Advanced methods for prototype-based classification

Schneider, Petra

Material based on:

Petra Schneider, Michael Biehl and Barbara Hammer - *"Distance Learning in Discriminative Vector Quantization,"* Neural Computation, vol. 21, no. 10, 2009.

Petra Schneider, Michael Biehl and Barbara Hammer - *"Adaptive Relevance Matrices in Learning Vector Quantization,"* Neural Computation, vol. 21, no. 12, 2009.

Chapter 3

Matrix learning in LVQ

Abstract

Learning vector quantization and extensions thereof offer efficient and intuitive classifiers which are based on the representation of classes by prototypes. The original methods, however, rely on the Euclidean distance corresponding to the assumption that the data can be represented by isotropic clusters. For this reason, extensions of the methods to more general metric structures have been proposed such as relevance adaptation in generalized LVQ (GLVQ) and robust soft LVQ (RSLVQ). In these approaches, metric parameters are learned based on the given classification task such that a data driven distance measure is found. We consider full matrix adaptation in advanced LVQ schemes; in particular, we introduce matrix learning to a recent statistical formalization of LVQ, robust soft LVQ, and we compare the results on several artificial and real life data sets to matrix learning in GLVQ, which is a derivation of LVQ-like learning based on a (heuristic) cost function. In all cases, matrix adaptation allows a significant improvement of the classification accuracy. Interestingly, however, the principled behavior of the models with respect to prototype locations and extracted matrix dimensions shows several characteristic differences depending on the data sets.

3.1 Introduction

Discriminative vector quantization schemes such as learning vector quantization (LVQ) are very popular classification methods due to their intuitivity and robustness: they represent the classification by (usually few) prototypes which constitute typical representatives of the respective classes and, thus, allow a direct inspection of the given classifier. All the methods presented in Sec. 2.3, however, suffer from the problem that classification is based on a predefined metric. The use of Euclidean distance, for instance, corresponds to the implicit assumption of isotropic clusters.

Such models can only be successful if the data displays a Euclidean characteristic. This is particularly problematic for high-dimensional data where noise accumulates and disrupts the classification, or heterogeneous data sets where different scaling and correlations of the dimensions can be observed. Thus, a more general metric structure would be beneficial in such cases. The field of metric adaptation constitutes a very active research topic in various distance based approaches such as unsupervised or semi-supervised clustering and visualization (Arnonkijpanich et al., 2008; Kaski, 2001), k-nearest neighbor approaches (Strickert et al., 2007; Weinberger et al., 2006) and learning vector quantization (Hammer and Villmann, 2002; Schneider et al., 2009a). We will focus on matrix learning in LVQ schemes which accounts for pairwise correlations of features, i.e. a very general and flexible set of classifiers. On the one hand, we will investigate the behavior of generalized matrix LVQ (GMLVQ) in detail, a matrix adaptation scheme for GLVQ which is based on a heuristic, though intuitive cost function. On the other hand, we will develop matrix adaptation for RSLVQ, a statistical model for LVQ schemes, and thus we will arrive at a uniform statistical formulation for prototype and metric adaptation in discriminative prototype-based classifiers. We will introduce variants which adapt the matrix parameters globally based on the training set or locally for every given prototype or mixture component, respectively.

Matrix learning in GLVQ and RSLVQ will be evaluated and compared in different learning scenarios: first, we consider test scenarios where prior knowledge about the form of the data is available. Furthermore, we compare the methods on two benchmarks from the UCI repository (Newman et al., 1998).

Interestingly, depending on the data, the methods show different characteristic behavior with respect to prototype locations and learned metrics. Although the classification accuracy is in many cases comparable, they display quite different behavior concerning their robustness with respect to parameter choices and the characteristics of the solutions. We will point out that these findings have consequences on the interpretability of the results. In all cases, however, matrix adaptation leads to an improvement of the classification accuracy, despite a largely increased number of free parameters.

3.2 Advanced distance measure

We introduce an important extension of the concepts presented in Sec. 2.4, which employs a full matrix of adaptive relevances in the similarity measure. We consider a generalized distance of the form

$$d^{\Lambda}(\boldsymbol{\xi}, \boldsymbol{w}) = (\boldsymbol{\xi} - \boldsymbol{w})^{\top} \Lambda (\boldsymbol{\xi} - \boldsymbol{w}), \tag{3.1}$$

where Λ is a full $n \times n$ matrix which can account for correlations between the features. For Λ to define a valid metric, symmetry and positive definiteness has to be enforced (we will discuss in a moment how this property can be guaranteed). This way, arbitrary Euclidean metrics can be realized by an appropriate choice of the parameters. In particular, correlations of dimensions and rotation of the axes can be accounted for. Such choices have already successfully been introduced in unsupervised clustering methods such as fuzzy clustering (Gath and Geva, 1989; Gustafson and Kessel, 1979), however, at the expense of increased computational costs, since these methods require a matrix inversion at each adaptation step. For the metric as introduced above, a variant which costs $\mathcal{O}(n^2)$ can be derived.

Note that, as already stated, the above similarity measure defines a general squared Euclidean distance in an appropriately transformed space only if Λ is positive definite and symmetric. We can achieve this by substituting

$$\Lambda = \Omega^\top \Omega \tag{3.2}$$

which yields $\boldsymbol{u}^\top \Lambda \boldsymbol{u} = \boldsymbol{u}^\top \Omega^\top \Omega \boldsymbol{u} = \left(\Omega^\top u\right)^2 \geq 0$ for all $\boldsymbol{u}$, where Ω is an arbitrary real $m \times n$ matrix, with $m \leq n$. In this chapter, we focus on the special case $m = n$. In addition, $\det \Lambda \neq 0$ has to be enforced to guarantee that Λ is positive definite. However, in practice, positive semi-definiteness of the matrix is sufficient, since data often only populates a sub-manifold of the full data space and definiteness has to hold only with regard to the relevant subspace of data. Therefore, we do not enforce $\det \Lambda \neq 0$. Using the relation in Eq. (3.2), the squared distance reads

$$d^\Lambda(\boldsymbol{\xi}, \boldsymbol{w}) = \sum_{ijk} (\xi^i - w^i) \Omega_{ki} \Omega_{kj} (\xi^j - w^j).$$

Ω can also be chosen to be symmetric, i.e. $\Omega^\top = \Omega$. In this case, the distance measure reads

$$d^\Lambda(\boldsymbol{\xi}, \boldsymbol{w}) = \sum_{ijk} (\xi^i - w^i) \Omega_{ik} \Omega_{kj} (\xi^j - w^j),$$

but we consider the more general case of non-symmetric Ω throughout the thesis.

Note that Ω realizes a linear transformation to an m-dimensional feature space. The metric d^Λ corresponds to the squared Euclidean distance in this new coordinate system. This can be seen by rewriting Eq. (3.1) as follows:

$$d^\Lambda(\boldsymbol{w}, \boldsymbol{\xi}) = \left[(\boldsymbol{\xi} - \boldsymbol{w})^\top \Omega^\top\right] \left[\Omega(\boldsymbol{\xi} - \boldsymbol{w})\right] = \left[\Omega(\boldsymbol{\xi} - \boldsymbol{w})\right]^2.$$

Hence, using the novel distance measure, the LVQ classifier is not restricted to the original features any more to classify the data. The system is able to detect alternative directions in feature space which provide more discriminative power to separate the classes. Choosing $m < n$ implies that the classifier is restricted to a reduced

number of features compared to the original input dimensionality of the data. Consequently, $\mathrm{rank}(\Lambda) \leq m$ and at least $n - m$ eigenvalues of Λ are equal to zero. In many applications, the intrinsic dimensionality of the data is smaller than the original number of features. Hence, this approach does not necessarily constrict the performance of the classifier extensively.

Since the optimal matrices Λ and Ω are not known in advance, they have to be learned from example data during training. To obtain the adaptation formulas for the different learning algorithms, we need to compute the derivatives with respect to $\boldsymbol{w}$ and Ω. The derivatives of $d^{\Lambda}(\cdot,\cdot)$ with respect to $\boldsymbol{w}$ and a single element Ω_{lm} yield

$$\frac{\partial d^{\Lambda}(\boldsymbol{\xi},\boldsymbol{w})}{\partial \boldsymbol{w}} = -2\,\Omega^{\top}\Omega\,(\boldsymbol{\xi}-\boldsymbol{w}) = -2\,\Lambda\,(\boldsymbol{\xi}-\boldsymbol{w}), \tag{3.3}$$

$$\frac{\partial d^{\Lambda}(\boldsymbol{\xi},\boldsymbol{w})}{\partial \Omega_{lm}} = 2\sum_i (\xi^i - w^i)\Omega_{li}(\xi^m - w^m). \tag{3.4}$$

After every learning step, Λ needs to be normalized to prevent the learning algorithm from degeneration. One possibility is to enforce

$$\sum_i \Lambda_{ii} = 1, \tag{3.5}$$

by dividing all elements of Λ by the *raw* value of $\sum_i \Lambda_{ii}$ after each step. In this way we fix the sum of diagonal elements which coincides with the sum of eigenvalues. This generalizes the normalization of relevances $\sum_i \lambda_i = 1$ for a simple diagonal metric. One can interpret the eigendirections of Λ as the temporary coordinate system with the relevances corresponding to the eigenvalues. Since

$$\Lambda_{ii} = \sum_k \Omega_{ki}\Omega_{ki} = \sum_k (\Omega_{ki})^2, \tag{3.6}$$

normalization can be done after every update step by dividing all elements of Ω by $(\sum_{ki}(\Omega_{ki})^2)^{1/2} = \left(\sum_i [\Omega^{\top}\Omega]_{ii}\right)^{1/2}$.

We can work with one full matrix which accounts for a transformation of the whole input space, or, alternatively, with local matrices attached to the individual prototypes. In the latter case, the squared distance of data point $\boldsymbol{\xi}$ from a prototype $\boldsymbol{w}_j$ is computed as

$$d^{\Lambda_j}(\boldsymbol{w}_j,\boldsymbol{\xi}) = (\boldsymbol{\xi}-\boldsymbol{w}_j)^{\top}\Lambda_j(\boldsymbol{\xi}-\boldsymbol{w}_j). \tag{3.7}$$

Each matrix is adapted individually. Localized matrices have the potential to take into account correlations which can vary between different classes or regions in feature space. For instance, clusters with ellipsoidal shape and different orientation

could be present in the data. Note that local matrices imply nonlinear decision boundaries which are composed of quadratic pieces, unlike a global matrix which is characterized by piecewise linear decision boundaries. This way, the receptive fields of the prototypes need no longer be convex or even connected, as we will see in the experiments. Depending on the data at hand, this effect can largely increase the capacity of the system.

3.3 Learning algorithms

In the following, we derive the update rules for matrix learning in LVQ1, Generalized LVQ and Robust Soft LVQ.

3.3.1 Matrix LVQ1

Matrix LVQ1 (MLVQ1) is the heuristic extension of LVQ1 as described in Sec. 2.3.1. The algorithm realizes Hebbian updates for the closest prototype $\boldsymbol{w}_L$ and the metric parameters. Accordingly, a learning step in MLVQ1 succeeds the following procedure:

1. Randomly select a training sample $(\boldsymbol{\xi}, y)$
2. Determine the winning prototype $\boldsymbol{w}_L$ with $d^{\Lambda}(\boldsymbol{w}_L, \boldsymbol{\xi}) = \min_l \{d^{\Lambda}(\boldsymbol{w}_l, \boldsymbol{\xi})\}$, breaking ties arbitrarily
3. Update $\boldsymbol{w}_L$ according to

$$\begin{aligned} \boldsymbol{w}_L &\leftarrow \boldsymbol{w}_L + \alpha_1 \cdot \Lambda \cdot (\boldsymbol{\xi} - \boldsymbol{w}_L), && \text{if } c(\boldsymbol{w}_L) = y, \\ \boldsymbol{w}_L &\leftarrow \boldsymbol{w}_L - \alpha_1 \cdot \Lambda \cdot (\boldsymbol{\xi} - \boldsymbol{w}_L), && \text{if } c(\boldsymbol{w}_L) \neq y \end{aligned} \tag{3.8}$$

4. Update Ω according to

$$\begin{aligned} \Omega &\leftarrow \Omega - \alpha_2 \cdot \Omega \cdot (\boldsymbol{\xi} - \boldsymbol{w}_L)(\boldsymbol{\xi} - \boldsymbol{w}_L)^{\top}, && \text{if } c(\boldsymbol{w}_L) = y, \\ \Omega &\leftarrow \Omega + \alpha_2 \cdot \Omega \cdot (\boldsymbol{\xi} - \boldsymbol{w}_L)(\boldsymbol{\xi} - \boldsymbol{w}_L)^{\top}, && \text{if } c(\boldsymbol{w}_L) \neq y, \end{aligned} \tag{3.9}$$

followed by a normalization step, and α_2 is the learning rate for the metric parameters. The matrix $\Lambda = \Omega^{\top}\Omega$ is updated in such a way that the distance $d^{\Lambda}(\boldsymbol{\xi}, \boldsymbol{w}_L)$ is decreased in case of a correct classification, while $d^{\Lambda}(\boldsymbol{\xi}, \boldsymbol{w}_L)$ increases, if the sample $(\boldsymbol{\xi}, y)$ is misclassified.

3.3.2 Generalized Matrix LVQ

To extend GLVQ with respect to the generalized distance measure, we replace the squared Euclidean distance in Eq. (2.7) by the novel metric introduced in Eq. (3.1). The newly derived cost function will be called E_{GMLVQ}:

$$E_{\mathrm{GMLVQ}} = \sum_{i=1}^{P} \Phi(\mu_i^{\Lambda}), \quad \text{with} \quad \mu_i^{\Lambda} = \frac{d_J^{\Lambda}(\boldsymbol{\xi}_i) - d_K^{\Lambda}(\boldsymbol{\xi}_i)}{d_J^{\Lambda}(\boldsymbol{\xi}_i) + d_K^{\Lambda}(\boldsymbol{\xi}_i)}, \tag{3.10}$$

where $d_J^{\Lambda}(\boldsymbol{\xi}) = d^{\Lambda}(\boldsymbol{w}_J, \boldsymbol{\xi})$ and $d_K^{\Lambda}(\boldsymbol{\xi}) = d^{\Lambda}(\boldsymbol{w}_K, \boldsymbol{\xi})$ constitute the distances of pattern $(\boldsymbol{\xi}, y)$ to the respective closest correct and wrong prototype. The derivatives of E_{GMLVQ} with respect to $\boldsymbol{w}_J$ and $\boldsymbol{w}_K$ and Ω_{lm} (see Sec. 3.A.1) and the derivatives in Eq.s (3.3) and (3.4) yield the updates for the prototypes and the metric parameters

$$\Delta \boldsymbol{w}_J = +\alpha_1 \cdot \Phi'(\mu^{\Lambda}(\boldsymbol{\xi})) \cdot \mu_J^{\Lambda}(\boldsymbol{\xi}) \cdot \Lambda \cdot (\boldsymbol{\xi} - \boldsymbol{w}_J), \tag{3.11}$$

$$\Delta \boldsymbol{w}_K = -\alpha_1 \cdot \Phi'(\mu^{\Lambda}(\boldsymbol{\xi})) \cdot \mu_K^{\Lambda}(\boldsymbol{\xi}) \cdot \Lambda \cdot (\boldsymbol{\xi} - \boldsymbol{w}_K), \tag{3.12}$$

$$\begin{aligned} \Delta \Omega_{lm} = &-\alpha_2 \cdot \Phi'(\mu^{\Lambda}(\boldsymbol{\xi})) \cdot \\ &\Big(\mu_J^{\Lambda}(\boldsymbol{\xi}) \cdot \Big((\xi_m - w_{J,m}) \left[\Omega(\boldsymbol{\xi} - \boldsymbol{w}_J)\right]_l \Big) - \\ &\mu_K^{\Lambda}(\boldsymbol{\xi}) \cdot \Big((\xi_m - w_{K,m}) \left[\Omega(\boldsymbol{\xi} - \boldsymbol{w}_K)\right]_l \Big) \Big), \end{aligned} \tag{3.13}$$

where $\mu_J^{\Lambda}(\boldsymbol{\xi}) = 4\, d_K^{\Lambda}(\boldsymbol{\xi}) / (d_J^{\Lambda}(\boldsymbol{\xi}) + d_K^{\Lambda}(\boldsymbol{\xi}))^2$ and $\mu_K^{\Lambda}(\boldsymbol{\xi}) = 4\, d_J^{\Lambda}(\boldsymbol{\xi}) / (d_J^{\Lambda}(\boldsymbol{\xi}) + d_K^{\Lambda}(\boldsymbol{\xi}))^2$.

Note that these updates correspond to the standard Hebb terms, pushing the closest correct prototype towards the considered data point and the closest wrong prototype away from the considered data point. The same holds for the update of the metric parameters, since the driving force consists of the derivative of the distance from the closest correct prototype (scaled with -1) and the closest incorrect prototype. Thus, the parameters of the matrix are changed in such a way that the distance from the closest correct prototype becomes smaller, whereas the distance from the closest wrong prototype is increased. We name the algorithm defined by Eq.s (3.11) - (3.13) Generalized Matrix LVQ (GMLVQ).

In the experimental section, we also derive local relevance matrices attached to each prototype. Each matrix is adapted individually in the following way: given $(\boldsymbol{\xi}, y)$ with closest correct prototype $\boldsymbol{w}_J$ and closest incorrect prototype $\boldsymbol{w}_K$, the up-

dates yield

$$\begin{aligned}\Delta\Omega_{J,lm} &= -\alpha_2 \cdot \Phi'(\mu^{\Lambda}(\boldsymbol{\xi})) \cdot \\ &\qquad \mu_J^{\Lambda}(\boldsymbol{\xi}) \cdot \Big((\xi_m - w_{J,m})[\Omega_J(\boldsymbol{\xi} - \boldsymbol{w}_J)]_l\Big), \\ \Delta\Omega_{K,lm} &= +\alpha_2 \cdot \Phi'(\mu^{\Lambda}(\boldsymbol{\xi})) \cdot \\ &\qquad \mu_K^{\Lambda}(\boldsymbol{\xi}) \cdot \Big((\xi_m - w_{K,m})[\Omega_K(\boldsymbol{\xi} - \boldsymbol{w}_K)]_l\Big)\,.\end{aligned} \tag{3.14}$$

Consequently, the update rules for the prototypes (Eq.s (3.11),(3.12)) also contain the local matrices Λ_J, Λ_K. We refer to this generalized version of GMLVQ as localized GMLVQ (LGMLVQ).

3.3.3 Matrix Robust Soft LVQ

To extend RSLVQ by the more general metric $d^{\Lambda}(\cdot,\cdot)$, the conditional density function $p(\boldsymbol{\xi}|j)$ in Eq. (2.10) needs to be defined in terms of

$$f(\boldsymbol{\xi}, \boldsymbol{w}, \sigma^2, \Omega) = \frac{-(\boldsymbol{\xi} - \boldsymbol{w})^{\top}\Omega^{\top}\Omega(\boldsymbol{\xi} - \boldsymbol{w})}{2\sigma^2}, \tag{3.15}$$

with

$$\frac{\partial f(\boldsymbol{\xi}, \boldsymbol{w}, \sigma^2, \Omega)}{\partial \boldsymbol{w}} = \frac{1}{\sigma^2}\,\Omega^{\top}\Omega\,(\boldsymbol{\xi} - \boldsymbol{w}) = \frac{1}{\sigma^2}\,\Lambda\,(\boldsymbol{\xi} - \boldsymbol{w}), \tag{3.16}$$

$$\frac{\partial f(\boldsymbol{\xi}, \boldsymbol{w}, \sigma^2, \Omega)}{\partial \Omega_{lm}} = -\frac{1}{\sigma^2}\,\left(\sum_i (\xi_i - w_i)\,\Omega_{li}\,(\xi_m - w_m)\right). \tag{3.17}$$

Substituting $f(\boldsymbol{\xi}, \boldsymbol{w}, \sigma^2, \Omega)$ in Eq. (2.12) yields the cost function of the novel algorithm Matrix Robust Soft LVQ (MRSLVQ). The cost function is maximized by means of a stochastic gradient ascent. The derivative of E_{MRSLVQ} with respect to the model parameters is stated in the appendix (see Sec. 3.A.2). Eq. (3.27) in combination with Eq.s (3.16) and (3.17) yields the update rules for the prototypes and the metric parameters

$$\Delta\boldsymbol{w}_j = \frac{\alpha_1}{\sigma^2}\begin{cases} (P_y(j|\boldsymbol{\xi}) - P(j|\boldsymbol{\xi}))\cdot\Lambda\cdot(\boldsymbol{\xi} - \boldsymbol{w}_j), & c(\boldsymbol{w}_j) = y, \\ -P(j|\boldsymbol{\xi})\cdot\Lambda\cdot(\boldsymbol{\xi} - \boldsymbol{w}_j), & c(\boldsymbol{w}_j) \neq y, \end{cases} \tag{3.18}$$

$$\Delta\Omega_{lm} = -\frac{\alpha_2}{\sigma^2} \cdot \sum_j \left(\Big(\delta_{y,c(\boldsymbol{w}_j)} \left(P_y(j|\boldsymbol{\xi}) - P(j|\boldsymbol{\xi})\right) - \left(1 - \delta_{y,c(\boldsymbol{w}_j)}\right) P(j|\boldsymbol{\xi}) \Big) \cdot \Big([\Omega(\boldsymbol{\xi} - \boldsymbol{w}_j)]_l \left(\xi_m - w_{j,m}\right) \Big) \right). \tag{3.19}$$

The assignment probabilities $P_y(j|\boldsymbol{\xi})$ and $P(j|\boldsymbol{\xi})$ are defined in Eq.s (3.30) and (3.31).

Similar to local matrix learning in GMLVQ, it is also possible to train an individual matrix Λ_j for every prototype. With individual matrices attached to all prototypes, the modification of Eq. (3.18) which includes the local matrices Λ_j is accompanied by

$$\Delta\Omega_{j,lm} = -\frac{\alpha_2}{\sigma^2} \cdot \left[\Big(\delta_{y,c(\boldsymbol{w}_j)} (P_y(j|\boldsymbol{\xi}) - P(j|\boldsymbol{\xi})) - (1 - \delta_{y,c(\boldsymbol{w}_j)}) P(j|\boldsymbol{\xi}) \Big) \cdot \Big([\Omega_j(\boldsymbol{\xi} - \boldsymbol{w}_j)]_l \left(\xi_m - w_{j,m}\right) \Big) \right], \tag{3.20}$$

under the constraint $K(j) = const.$ for all j. We term this learning rule localized MRSLVQ (LMRSLVQ). Due to the restriction to constant normalization factors $K(j)$, the normalization $\det(\Lambda_j) = const.$ is assumed for this algorithm.

In the following experiments, we determine the algorithm's hyperparpameter σ^2 by means of a validation procedure. Alternatively, the parameter can be learned from the data which will be introduced in Sec. 6.2.

3.4 Experiments

With respect to parameter initialization and learning rate annealing, we use the same procedures in all experiments. The mean values of random subsets of training samples selected from each class are chosen as initial states of the prototypes. The learning rates are continuously reduced in the course of learning. Following Darken et al. (1992), we implement a schedule of the form

$$\alpha_i(t) = \frac{\alpha_i}{1 + \tau\,(t-1)} \tag{3.21}$$

($i \in \{1, 2\}$), where t counts the number training epochs. The factor τ determines the speed of annealing and is chosen individually for every application. The hyperparameter σ^2 is held constant in all experiments with RSLVQ and MRSLVQ. To

normalize the relevance matrices after each learning step, we choose $\sum_i \Lambda_{ii} = 1$ and initially set $\Lambda = \mathbb{1}/n$. Note that, in consequence, the Euclidean distance in RSLVQ and GLVQ has to be normalized to one as well to allow for a fair comparison with respect to learning rates. Accordingly, the RSLVQ- and GLVQ prototype updates and the function f in Eq. (2.9) have to be weighted by $1/n$.

3.4.1 Artificial data

In the first experiments, the algorithms are applied to the artificial data from Bojer et al. (2001) to illustrate the training of an LVQ-classifier based on the alternative cost functions with fixed and adaptive distance measure. The data sets 1 and 2 comprise three-class classification problems in a two dimensional space. Each class is split into two clusters with small or large overlap, respectively (see Fig. 3.1). We randomly select 2/3 of the data samples of each class for training and use the remaining data for testing. According to the a priori known distributions, the data is represented by two prototypes per class. We use the learning parameter settings

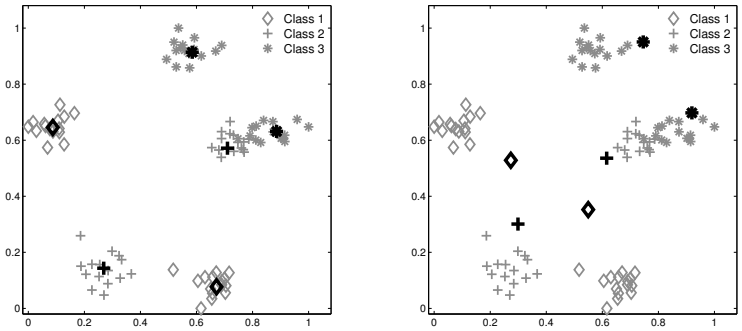
G(M)LVQ: $4\,\alpha_1 = 0.005,\ \ 4\,\alpha_2 = 0.001$
(M)RSLVQ: $\alpha_1 = 5 \cdot 10^{-4},\ \alpha_2 = 1 \cdot 10^{-4}$

$\tau = 0.001$ and perform 1000 sweeps through the training set. The results presented in the following are averaged over 10 independent constellations of training and test set. We apply several different values σ^2 from the interval [0.001, 0.015] and present the simulations giving rise to the best mean performance on the training sets.

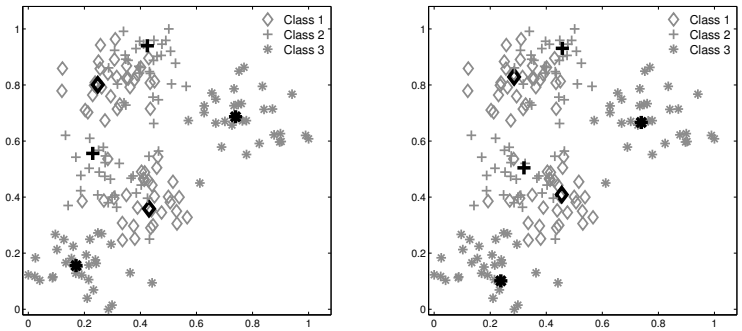
The results are summarized in Tab. 3.1. They are obtained with the hyperparameters $\sigma^2_{opt}(\text{RSLVQ}) = 0.002$ and $\sigma^2_{opt}(\text{MRSLVQ}) = 0.002,\ 0.003$ for data set 1 and 2, respectively. The use of the advanced distance measure yields only a slight improvement compared to the fixed Euclidean distance, since the distributions do not

Table 3.1: *Mean rate of misclassification (in %) obtained by the different algorithms on the artificial data sets 1 and 2 at the end of training. The values in brackets are the variances.*

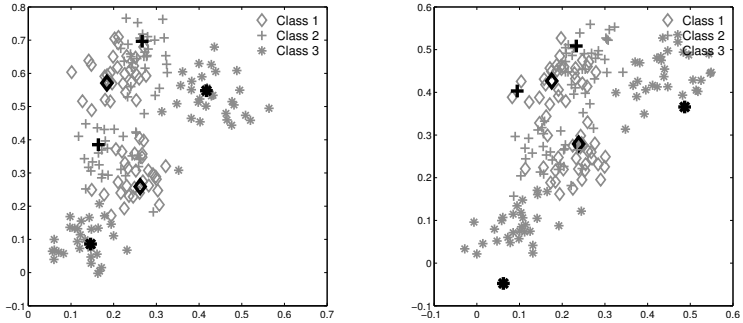
	Data set 1		Data set 2	
Algorithm	ε_{train}	ε_{test}	ε_{train}	ε_{test}
GLVQ	2.0 (0.02)	2.7 (0.07)	19.2 (0.9)	24.2 (1.9)
GMLVQ	2.0 (0.02)	2.7 (0.07)	18.6 (0.7)	23.0 (1.6)
RSLVQ	1.5 (0.01)	3.7 (0.04)	12.8 (0.07)	19.3 (0.3)
MRSLVQ	1.5 (0.01)	3.7 (0.02)	12.3 (0.04)	19.3 (0.3)



(a) Data set 1. Left: GLVQ prototypes. Right: RSLVQ prototypes

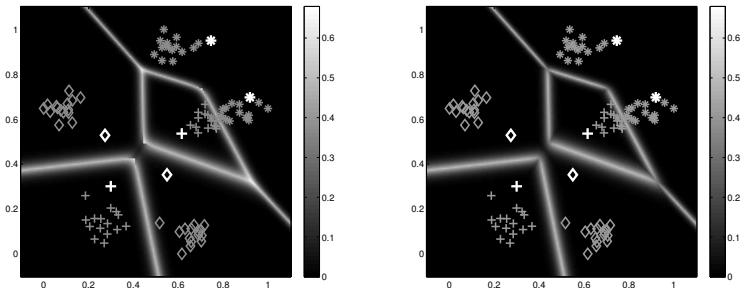


(b) Data set 2. Left: GLVQ prototypes. Right: RSLVQ prototypes

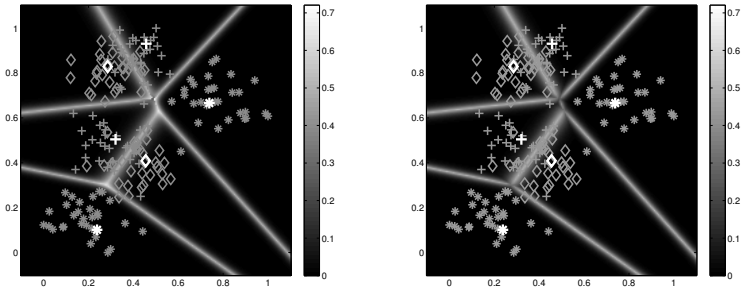


(c) Data set 3. Left: GMLVQ prototypes. Right: MRSLVQ prototypes. The plots relate to the first two dimensions after projecting the data and the prototypes with Ω_{GMLVQ} and Ω_{MRSLVQ}, respectively.

Figure 3.1: Artificial data. *Prototype constellations identified by GLVQ, RSLVQ, GMLVQ and MRSLVQ in a single run on different data sets.*



(a) Data set 1. Left: Attractive forces. Right: Repulsive forces. The plots relate to the hyperparameter $\sigma^2 = 0.002$.



(b) Data set 2. Left: Attractive forces. Right: Repulsive forces. The plots relate to the hyperparameter $\sigma^2 = 0.002$.

Figure 3.2: Artificial data. *Visualization of the update factors* $(P_y(j|\boldsymbol{\xi}) - P(j|\boldsymbol{\xi}))$ *(attractive forces) and* $P(j|\boldsymbol{\xi})$ *(repulsive forces) of the nearest prototype with correct and incorrect class label on data sets 1 and 2.*

have favorable directions to classify the data. On data set 1, GLVQ and RSLVQ show nearly the same performance. However, the prototype configurations identified by the two algorithms vary significantly (see Fig. 3.1a). During GLVQ-training, the prototypes move close to the cluster centers in only a few training epochs, resulting in an appropriate approximation of the data by the prototypes. On the contrary, prototypes are frequently located outside the clusters, if the classifier is trained with the RSLVQ-algorithm. This behavior is due to the fact that only data points lying close to the decision boundary change the prototype constellation in RSLVQ significantly (see Eq. (3.18)). As depicted in Fig. 3.2a, only a small number of training samples are lying in the active region of the prototypes while the great majority of training samples attains only tiny weight values which are not sufficent to adjust the prototypes to the data in reasonable training time. This effect does not have negative

impact on the classification of the data set. However, the prototypes do not provide a reasonable approximation of the data.

The prototype constellation identified by RSLVQ on data set 2 represents the classes clearly better (see Fig. 3.1b). Since the clusters show significant overlap, a sufficiently large number of training samples contributes to the learning process (see Fig. 3.2b) and the prototypes quickly adapt to the data. The good approximation of the data is accompanied by an improved classification performance compared to GLVQ. Although GLVQ also places prototypes close to the cluster centers, the use of the RSLVQ-cost function gives rise to the superior classifier for this data set. This observation is also confirmed by the experiments with GMLVQ and MRSLVQ.

To demonstrate the influence of metric learning, data set 3 is generated by embedding each sample $\boldsymbol{\xi} = (\xi_1, \xi_2) \in \mathbb{R}^2$ of data set 2 in $\mathbb{R}^{10}$ by choosing: $\xi_3 = \xi_1 + \eta_1, \ldots \xi_6 = \xi_1 + \eta_4$, where η_i comprises Gaussian noise with variances 0.05, 0.1, 0.2 and 0.5, respectively. The features $\xi_7, \ldots, \xi_{10}$ contain pure uniformly distributed noise in [-0.5, 0.5] and [-0.2, 0.2] and Gaussian noise with variances 0.5 and 0.2, respectively. Hence, the first two dimensions are most informative to classify this data set. The dimensions 3 to 6 still partially represent dimension 1 with increasing noise added. Finally, we apply a random linear transformation on the samples of data set 3 in order to construct a test scenario, where the discriminating structure is not in parallel to the original coordinate axis any more. We refer to this data as data set 4. To train the classifiers for the high-dimensional data sets we use the same learning parameter settings as in the previous experiments.

The obtained mean rates of misclassification are reported in Tab. 3.2. The results are achieved using the hyperparameter settings $\sigma^2_{opt}(\text{RSLVQ}) = 0.002, 0.003$ and $\sigma^2_{opt}(\text{MRSLVQ}) = 0.003$ for data set 3 and 4, respectively. The performance of GLVQ

Table 3.2: *Mean rate of misclassification (in %) obtained by the different algorithms on the artificial data sets 3 and 4 at the end of training. The values in brackets are the variances.*

	Data set 3		Data set 4	
Algorithm	ε_{train}	ε_{test}	ε_{train}	ε_{test}
GLVQ	23.5 (0.1)	38.0 (0.2)	31.2 (0.1)	41.0 (0.2)
GMLVQ	12.1 (0.1)	24.0 (0.4)	14.5 (0.1)	30.6 (0.5)
RSLVQ	4.1 (0.1)	33.2 (0.5)	11.7 (0.1)	36.8 (0.2)
MRSLVQ	3.9 (0.1)	29.5 (0.4)	8.0 (0.1)	32.0 (0.2)

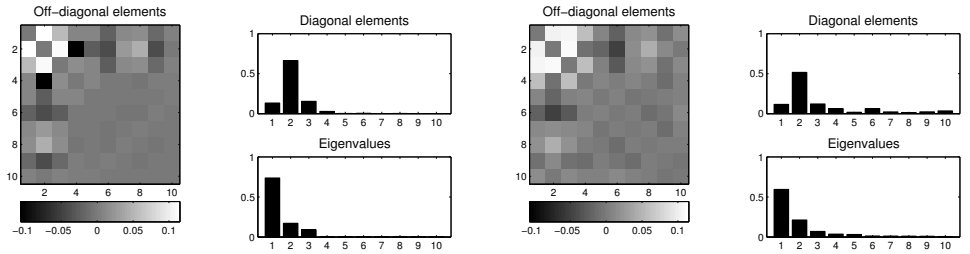


Figure 3.3: Artificial data. *Visualization of the relevance matrices* Λ_{GMLVQ} *(left) and* $\Lambda_{\mathrm{MRSLVQ}}$ *(right) obtained in one training run on data set 3. The diagonal elements are set to zero in the visualization of the off-diagonal elements.*

clearly degrades due to the additional noise in the data. However, by adapting the metric to the structure of the data, GMLVQ is able to achieve nearly the same accuracy on data sets 2 and 3. A visualization of the resulting relevance matrix Λ_{GMLVQ} is provided in Fig. 3.3, left. The diagonal elements turn out that the algorithm totally eliminates the noisy dimensions 4 to 10, which, in consequence, do not contribute to the computation of distances any more. As reflected by the off-diagonal elements, the classifier additionally takes correlations between the informative dimensions 1 to 3 into account to quantify the similarity of prototypes and feature vectors. Interestingly, the algorithms based on the statistically motivated cost function show strong overfitting effects on this data set. Obviously, the number of training examples in this application is not sufficient to allow for an unbiased estimation of the correlation matrices of the mixture model. MRSLVQ does not detect the relevant structure in the data sufficiently to reproduce the classification performance achieved on data set 2. The respective relevance matrix trained on data set 3 (see Fig. 3.3, right) depicts that the algorithm does not totally prune out the uninformative dimensions. The superiority of GMLVQ in this application is also reflected by the final position of the prototypes in feature space (see Fig. 3.1c). A comparable result for GMLVQ can even be observed after training the algorithm on data set 4. Hence, the method succeeds to detect the discriminative structure in the data, even after rotating or scaling the data arbitrarily.

3.4.2 Real life data

Beyond, we apply the algorithms to two benchmark data sets provided by the UCI repository of Machine Learning (Newman et al., 1998), namely Image Segmentation and Letter Recognition.

Image segmentation data set

The data set contains 19-dimensional feature vectors, which encode different attributes of 3×3 pixel regions extracted from outdoor images. Each such region is assigned to one of seven classes (brickface, sky, foliage, cement, window, path, grass). The features 3-5 are (nearly) constant and are eliminated for these experiments. As a further preprocessing step, the features are normalized to zero mean and unit variance. The training set contains 210 data points (30 samples per class), the test data consists of 300 samples per class. We split the test data into a test and a validation set of equal size in order to optimize the model settings.

Each class is approximated by one prototype, respectively. We use the parameters settings

G(M)LVQ: $4\,\alpha_1 = 0.005,\ 4\,\alpha_2 = 0.0001$
(M)RSLVQ: $\alpha_1 = 0.0001,\ \alpha_2 = 0.0001,\ \sigma^2 = 0.04$

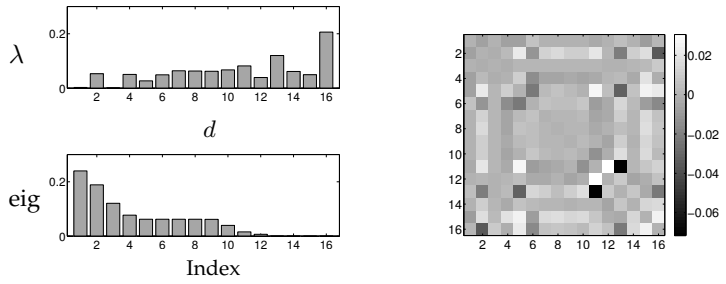
$\tau = 0.0001$ and continue learning until the validation error remains constant or starts indicating overfitting effects. In order to reduce the influence of random fluctuations, we average our results over ten runs with varying initializations.

The obtained classification accuracies are summarized in Table 3.3. For both cost function schemes, the performance improves significantly by adapting the distance measure. Remarkably, RSLVQ and MRSLVQ clearly outperform the respective GLVQ methods on this data set.

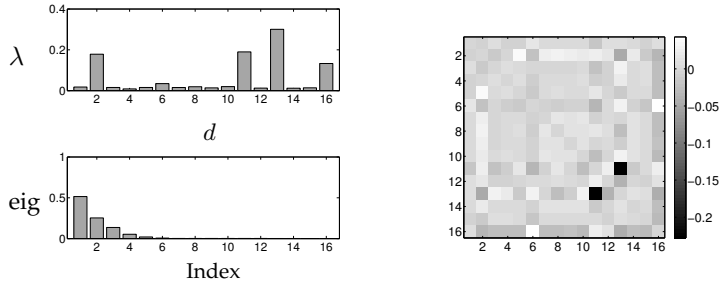
Comparing the resulting relevance matrices in Fig. 3.4 shows that GMLVQ and

Table 3.3: *Mean rate of misclassification (in %) obtained by the different algorithms on the image segmentation and letter data set at the end of training. The values in brackets constitute the variances.*

Algorithm	Image segmentation data		Letter data	
	ε_{train}	ε_{test}	ε_{train}	ε_{test}
GLVQ	15.2 (0.0)	17.0 (0.003)	28.4 (0.002)	28.9 (0.003)
GMLVQ	9.5 ($2 \cdot 10^{-5}$)	9.8 ($6 \cdot 10^{-5}$)	29.3 (0.002)	30.2 (0.002)
LGMLVQ	4.8 ($2 \cdot 10^{-4}$)	8.6 (0.004)	14.3 ($3 \cdot 10^{-4}$)	16.0 (0.002)
RSLVQ	7.1 ($2 \cdot 10^{-6}$)	9.4 ($2 \cdot 10^{-5}$)	21.9 (0.001)	23.2 (0.005)
MRSLVQ	1.7 ($1 \cdot 10^{-5}$)	6.2 ($2 \cdot 10^{-5}$)	21.7 (0.001)	22.9 (0.004)
LMRSLVQ	-	-	1.3 ($1 \cdot 10^{-4}$)	6.2 ($8 \cdot 10^{-4}$)



(a) Λ_{GMLVQ}



(b) $\Lambda_{\mathrm{MRSLVQ}}$

Figure 3.4: Image segmentation data. *Visualization of the relevance matrix Λ after 1500 epochs of GMLVQ-training and MRSLVQ-training.* **Left:** *Diagonal elements and eigenvalues.* **Right:** *Off-diagonal elements. The diagonal elements are set to zero for the plot.*

MRSLVQ identify the same dimensions as being most discriminative to classify the data. The features which achieve the highest weight values on the diagonal are the same in both cases. But note that the feature selection by MRSLVQ is clearly more pronounced. The eigenvalue profiles reflect that MRSLVQ uses a smaller number of features to classify the data. It is visible that especially relations between the dimensions encoding color information are emphasized. The dimensions weighted as most important are features 11: exred-mean (2R - (G + B)) and 13: exgreen-mean (2G - (R + B)) in both classes. Furthermore, the off-diagonal elements highlight correlations with e.g. feature 8: rawred-mean (average over the regions red values), feature 9: rawblue-mean (average over the regions green values), feature 10: rawgreen-mean (average over the regions green values). For a description of the features, see Newman et al. (1998).

Based on Λ_{GMLVQ}, distances between prototypes and feature vectors obtain much smaller values compared to the MRSLVQ-matrix. This is depicted in Fig. 3.7a which visualizes the distributions of the distances d_J^{Λ} and d_K^{Λ} to the closest correct and in-

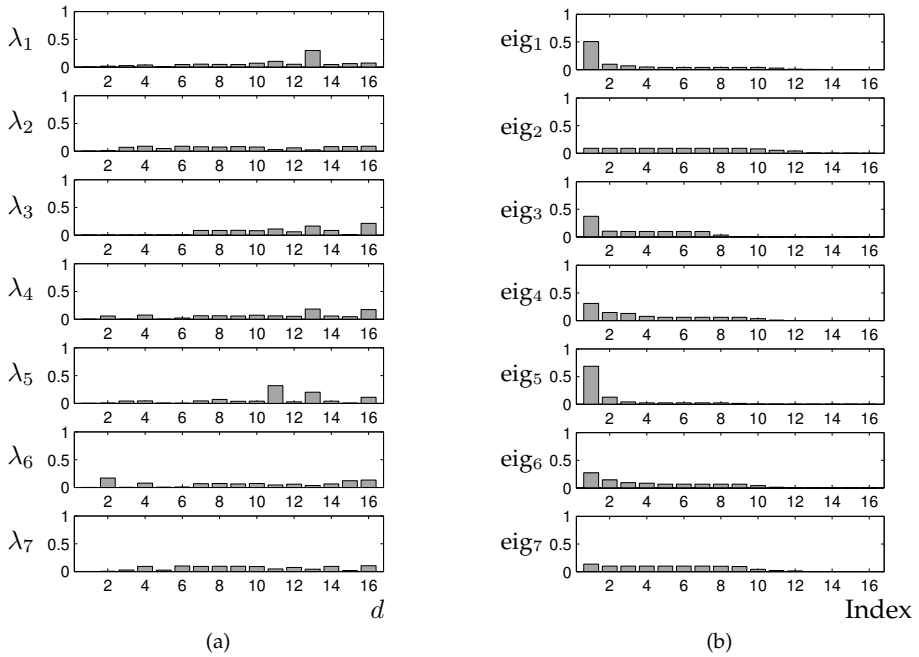
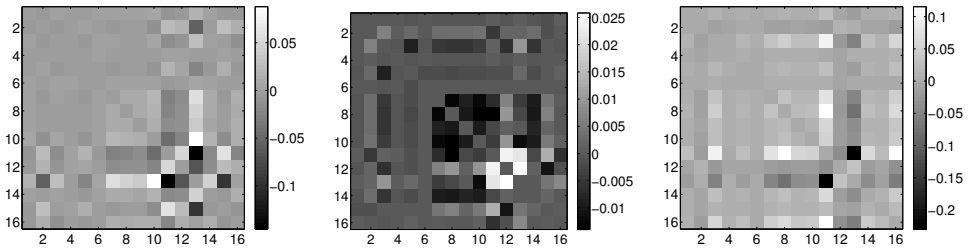


Figure 3.5: Image segmentation data. *(a) Diagonal elements of local relevance matrices* Λ_{1-7}. *(b) Eigenvalue spectra of local relevance matrices* Λ_{1-7}.

correct prototype. 90% of all test samples attain distances $d_J^\Lambda < 0.2$ by the GMLVQ classifiers. This holds for only 40% of the feature vectors, if the MRSLVQ classifiers are applied to the data. This observation is also reflected by the distribution of the data and the prototypes in the transformed feature spaces (see Fig. 3.8a). After projection with Ω_{GMLVQ} the data comprises very compact clusters with high density, while the samples and prototypes spread wider in the coordinate system detected by MRSLVQ.

In a further sequence of GMLVQ experiments, we analyse the algorithm's sensitivity with respect to the number of prototypes. We determine the class-wise rates of misclassification after each experiment and repeat the training providing one further prototype for the class contributing most to the overall rate of misclassification. A system consisting of 10 prototypes achieves 90.9% mean test accuracy, using 13 prototypes reaches 91.2% mean accuracy on the test sets. The slight improvement in classification performance is accompanied by an increasing training time until convergence.

Finally, we train Localized GMLVQ with identical learning parameters as in the

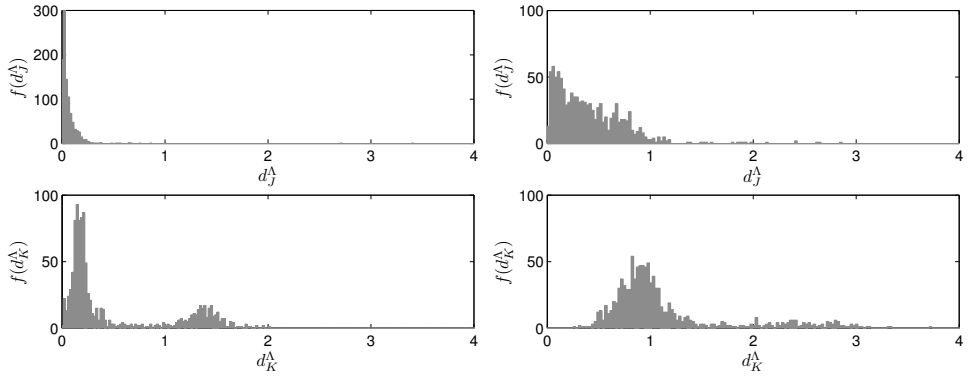


(a) Class 1: brickface (b) Class 2: sky (c) Class 5: window

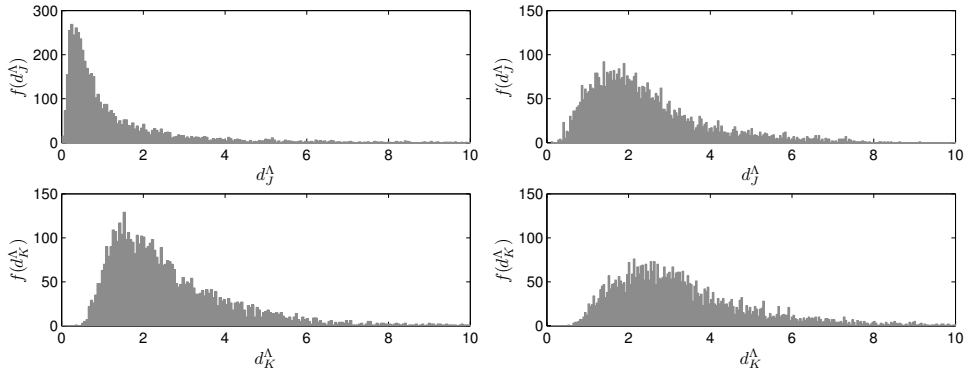
Figure 3.6: Image segmentation data. *Visualization of the off-diagonal elements of the local matrices $\Lambda_{1,2,5}$. The diagonal elements are set to zero for the plot.*

previous GMLVQ experiments. The training error remains constant after approximately 12 000 sweeps through the training set. The validation error shows slight over-fitting effects. It reaches a minimum after approximately 10 000 epochs and increases in the further course of training. In the following, we present the results obtained after 10 000 epochs. At this point, training of individual matrices per prototype achieves a test accuracy of 94.4%. We are aware of only one SVM result in the literature which is applicable for comparing the performance. Prudent and Ennaji (2005) achieve 93.95% accuracy on the test set.

Figure 3.5 shows the diagonal elements and eigenvalue spectra of all local matrices we obtain in one run which are also representative for the other experiments. Matrices with a clear preference for certain dimensions on the diagonal also display a distinct eigenvalue profile (e.g. Λ_1, Λ_5). Similarly, matrices with almost balanced relevance values on the diagonal exhibit only a weak decay from the first to the second eigenvalue (e.g. Λ_2, Λ_7). This observation for diagonal elements and eigenvalues coincides with a similar one for the off-diagonal elements. Figure 3.6 visualizes the off-diagonal elements of the local matrices Λ_1, Λ_2 and Λ_5. Corresponding to the balanced relevance- and eigenvalue profile of matrix Λ_2, the off-diagonal elements are only slightly different from zero. This may indicate diffuse data without a pronounced, hidden structure. There are obviously no other directions in feature space which could be used to significantly minimize distances within this class. On the contrary, the matrices Λ_1 and Λ_5 show a clearer structure. The off-diagonal elements cover a much wider range and there is a clearer emphasis on particular dimensions. This implies that class-specific correlations between certain features have significant influence. The most distinct weights for correlations with other dimensions are obtained for features, which also gain high relevance values on the diagonal.

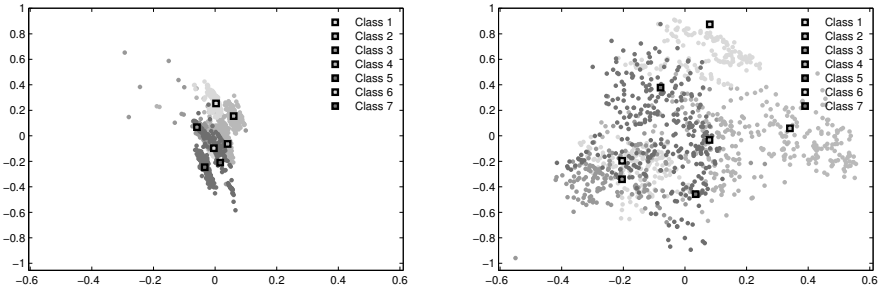


(a) Image segmentation data set. Left: Application of GMLVQ classifier. Right: Application of MRSLVQ classifier.

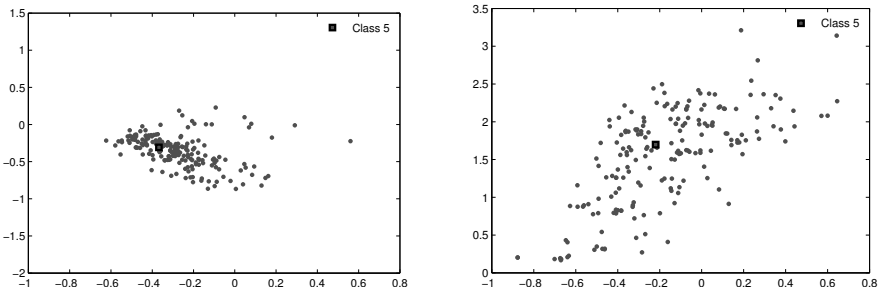


(b) Letter data set: Left: Application of Local GMLVQ classifier. Right: Application of Local MRSLVQ classifier. For this analysis, the matrices are normalized to $\sum_i \Lambda_{ii} = 1$ after training.

Figure 3.7: *Distributions of distances d_J^Λ and d_K^Λ of test data to closest correct and closest incorrect prototype; $f(d_{J,K}^\Lambda)$ specifies the absolute frequency of $d_{J,K}^\Lambda$.*



(a) Image segmentation data set. Left: Transformation with Ω_{GMLVQ}. Right: Transformation with Ω_{MRSLVQ}.



(b) Letter data set. Left: Transformation of class 5 data with $\Omega^5_{\mathrm{LGMLVQ}}$. Right: Transformation of class 5 data with $\Omega^5_{\mathrm{LMRSLVQ}}$. For these visualizations, the matrices are normalized to $\sum_{ij} \Omega^2_{ij} = 1$ after training.

Figure 3.8: *Data set visualizations with respect to the first two dimensions after projection with the global or local transformation matrices obtained by different training algorithms.*

Letter recognition data set

The data set consists of 20 000 feature vectors which encode 16 numerical attributes of black-and-white rectangular pixel displays of the 26 capital letters of the English alphabet. The features are scaled to fit into a range of integer values between 0 and 15. This data set is also used in Seo and Obermayer (2003) to analyse the performance of RSLVQ. We extract one half of the samples of each class for training the classifiers and one fourth for testing and validating, respectively. The following results are averaged over 10 independent constellations of the different data sets. We train the classifiers with one prototype per class respectively and use the learning parameter settings

(Local) G(M)LVQ: $4\,\alpha_1 = 0.05, 4\,\alpha_2 = 0.005$

(Local) (M)RSLVQ: $\alpha_1 = 0.01$, $\alpha_2 = 0.001$

and $\tau = 0.1$. Training is continued for 150 epochs in total with different values σ^2 lying in the interval [0.75, 4.0]. The accuracy on the validation set is used to select the best settings for the hyperparameter. With the settings $\sigma^2_{opt}(\text{RSLVQ}) = 1.0$ and $\sigma^2_{opt}(\text{MRSLVQ, Local MRSLVQ}) = 1.5$, we achieve the performances stated in Tab. 3.3. The results depict that training of an individual metric for every prototype is particularly efficient in case of multi-class problems. The adaptation of a global relevance matrix does not provide significant benefit because of the huge variety of classes in this application. Similar to the previous application, the RSLVQ-based algorithms outperform the methods based on the GLVQ cost function. The experiments also confirm our preceding observations regarding the distribution of distance values induced by the different relevance matrices. Since global matrix adaptation does not have significant impact on the classification performance, we relate to the simulations with Local GMLVQ and Local MRLSVQ in Fig. 3.7b. It depicts that the distances d^{Λ}_J and d^{Λ}_K assume larger values if the training is based on the RSLVQ cost function. Accordingly, the data distributions show similar characteristics as already described for the image segmentation data set after projection with $\Omega_{i,\text{LGMLVQ}}$ and $\Omega_{i,\text{LMRSLVQ}}$ (see Fig. 3.8b). Remarkably, the classification accuracy of Local MRSLVQ with one prototype per class is comparable to the RSLVQ results presented in Seo and Obermayer (2003), achieved with constant hyperparameter σ^2 and 13 prototypes per class. This observation underlines the crucial importance of an appropriate distance measure for the performance of LVQ-classifiers.

Despite the large number of parameters, we do not observe overfitting effects during training of local relevance matrices on this data set. The systems show stable behaviour and converge within 100 training epochs.

3.5 Conclusion

We have considered metric learning by matrix adaptation in discriminative vector quantization schemes. In particular, we have introduced this principle into robust soft learning vector quantization, which is based on an explicit statistical model by means of mixtures of Gaussians, and we compared this method to an alternative scheme derived from the intuitive but somewhat heuristic cost function of generalized learning vector quantization. In general, it can be observed that matrix adaptation allows to improve the classification accuracy on the one hand, and it leads to a simplification of the classifier and thus better interpretability of the results by inspection of the eigenvectors and eigenvalues on the other hand. Interestingly, the behavior of GMLVQ and MRSLVQ shows several principled differences. Based on

the experimental findings, the following conclusions can be drawn:

- All discriminative vector quantization schemes show good generalization behavior and yield reasonable classification accuracy on benchmark results using only few prototypes. RSLVQ seems particularly suited for the real-life data set considered in this section. In general, matrix learning allows to further improve the results, whereby, depending on the setting, overfitting can be more pronounced due to the huge number of free parameters.

- The methods are generally robust against noise in the data as can be inferred from different runs of the algorithm on different splits of the data sets. While GLVQ and variants are rather robust to the choice of hyperparameters, a very critical hyperparameter of training is the softness parameter σ^2 for RSLVQ. Matrix adaptation seems to weaken the sensitivity w.r.t. this parameter, however, a correct choice of σ^2 is still crucial for the classification accuracy and efficiency of the runs. For this reason, automatic adaptation schemes for σ^2 should be considered. In Seo and Obermayer (2006), a simple annealing scheme for σ^2 is introduced which yields reasonalbe results. A more principled way to adapt σ^2 according to the optimization of the likelihood ratio is presented in Chap. 6.

- The methods allow for an inspection of the classifier by means of the prototypes which are defined in input space. Note that one explicit goal of unsupervised vector quantization schemes such as k-means or the self-organizing map is to represent typical data regions be means of prototypes. Since the considered approaches are discriminative, it is not clear in how far this property is maintained for GLVQ and RSLVQ variants. The experimental findings demonstrate that GLVQ schemes place prototypes close to class centres and prototypes can be interpreted as typical class representatives. On the contrary, RSLVQ schemes do not preserve this property in particular for non-overlapping classes since adaptation basically takes place based on misclassifications of the data. Therefore, prototypes can be located outside the class centers while maintaining the same or a similar classification boundary compared to GLVQ schemes. This property has already been observed and proven in typical model situations using the theory of online learning for the limit learning rule of RSLVQ, learning from mistakes, in Biehl, Ghosh and Hammer (2007).

- Despite the fact that matrix learning introduces a huge number of additional free parameters, the method tends to yield very simple solutions which involve only few relevant eigendirections. This behavior can be substantiated

by an exact mathematical investigation of the LVQ2.1-type limit learning rules which result for small σ^2 or a steep sigmoidal function Φ, respectively. For these limits, an exact mathematical investigation becomes possible, indicating that a unique solution for matrix learning exist, given fixed prototypes, and that the limit matrix reduces to a singular matrix which emphasizes one major eigenvalue direction. The exact mathematical treatment of these simplified limit rules is subject of ongoing work and will be published in subsequent work.

In conclusion, systematic differences of GLVQ and RSLVQ schemes result from the different cost functions used in the approaches. This includes a larger sensitivity of RSLVQ to hyperparanmeters, a different location of prototypes which can be far from the class centres for RSLVQ, and different classification accuracies in some cases. Apart from these differences, matrix learning is clearly beneficial for both discriminative vector quantization schemes as demonstrated in the experiments.

3.A Appendix: Derivatives

3.A.1 E_{GMLVQ} with respect to $\boldsymbol{w}_{J,K}$ and Ω_{lm}

We assume the function $\Phi(x)$ in Eq. (3.10) to be the identity function $\Phi(x) = x$ which implies $\frac{\partial \Phi(x)}{\partial x} = 1$.

$$\frac{\partial E_{\mathrm{GMLVQ}}}{\partial \boldsymbol{w}_{J,K}} = \frac{\partial \Phi(\mu^{\Lambda}(\boldsymbol{\xi}))}{\partial \mu^{\Lambda}(\boldsymbol{\xi})} \cdot \frac{\partial \mu^{\Lambda}(\boldsymbol{\xi})}{\partial \boldsymbol{w}_{J,K}}, \tag{3.22}$$

$$\begin{aligned}\frac{\partial \mu^{\Lambda}(\boldsymbol{\xi})}{\partial \boldsymbol{w}_{J}} &= \frac{(d_J^{\Lambda}(\boldsymbol{\xi}) + d_K^{\Lambda}(\boldsymbol{\xi})) - (d_J^{\Lambda}(\boldsymbol{\xi}) - d_K^{\Lambda}(\boldsymbol{\xi}))}{(d_J^{\Lambda}(\boldsymbol{\xi}) + d_K^{\Lambda}(\boldsymbol{\xi}))^2} \cdot \frac{\partial d_J^{\Lambda}(\boldsymbol{\xi})}{\partial \boldsymbol{w}_{J}} \\ &= \frac{2 \cdot d_K^{\Lambda}(\boldsymbol{\xi})}{(d_J^{\Lambda}(\boldsymbol{\xi}) + d_K^{\Lambda}(\boldsymbol{\xi}))^2} \cdot \frac{\partial d_J^{\Lambda}(\boldsymbol{\xi})}{\partial \boldsymbol{w}_{J}},\end{aligned} \tag{3.23}$$

$$\begin{aligned}\frac{\partial \mu^{\Lambda}(\boldsymbol{\xi})}{\partial \boldsymbol{w}_{K}} &= -\frac{(d_J^{\Lambda}(\boldsymbol{\xi}) + d_K^{\Lambda}(\boldsymbol{\xi})) + (d_J^{\Lambda}(\boldsymbol{\xi}) - d_K^{\Lambda}(\boldsymbol{\xi}))}{(d_J^{\Lambda}(\boldsymbol{\xi}) + d_K^{\Lambda}(\boldsymbol{\xi}))^2} \cdot \frac{\partial d_K^{\Lambda}(\boldsymbol{\xi})}{\partial \boldsymbol{w}_{K}} \\ &= \frac{-2 \cdot d_J^{\Lambda}(\boldsymbol{\xi})}{(d_J^{\Lambda}(\boldsymbol{\xi}) + d_K^{\Lambda}(\boldsymbol{\xi}))^2} \cdot \frac{\partial d_K^{\Lambda}(\boldsymbol{\xi})}{\partial \boldsymbol{w}_{K}},\end{aligned} \tag{3.24}$$

$$\frac{\partial E_{\mathrm{GMLVQ}}}{\partial \Omega_{lm}} = \frac{\partial \Phi(\mu^{\Lambda}(\boldsymbol{\xi}))}{\partial \mu^{\Lambda}(\boldsymbol{\xi})} \cdot \frac{\partial \mu^{\Lambda}(\boldsymbol{\xi})}{\partial \Omega_{lm}}, \tag{3.25}$$

$$\begin{aligned}\frac{\partial \mu^{\Lambda}(\boldsymbol{\xi})}{\partial \Omega_{lm}} &= \frac{\left(\frac{\partial d_J^{\Lambda}(\xi)}{\partial \Omega_{lm}} - \frac{\partial d_K^{\Lambda}(\xi)}{\partial \Omega_{lm}}\right)(d_J^{\Lambda}(\boldsymbol{\xi}) + d_K^{\Lambda}(\boldsymbol{\xi}))}{(d_J^{\Lambda}(\boldsymbol{\xi}) + d_K^{\Lambda}(\boldsymbol{\xi}))^2} \\ &\quad - \frac{\left(\frac{\partial d_J^{\Lambda}(\xi)}{\partial \Omega_{lm}} + \frac{\partial d_K^{\Lambda}(\xi)}{\partial \Omega_{lm}}\right)(d_J^{\Lambda}(\boldsymbol{\xi}) - d_K^{\Lambda}(\boldsymbol{\xi}))}{(d_J^{\Lambda}(\boldsymbol{\xi}) + d_K^{\Lambda}(\boldsymbol{\xi}))^2} \\ &= \frac{2\, d_K^{\Lambda}(\boldsymbol{\xi})}{(d_J^{\Lambda}(\boldsymbol{\xi}) + d_K^{\Lambda}(\boldsymbol{\xi}))^2} \cdot \frac{\partial d_J^{\Lambda}(\boldsymbol{\xi})}{\partial \Omega_{lm}} - \frac{2\, d_J^{\Lambda}(\boldsymbol{\xi})}{(d_J^{\Lambda}(\boldsymbol{\xi}) + d_K^{\Lambda}(\boldsymbol{\xi}))^2} \cdot \frac{\partial d_K^{\Lambda}(\boldsymbol{\xi})}{\partial \Omega_{lm}}\end{aligned} \tag{3.26}$$

3.A.2 E_{MRSLVQ} with respect to general model parameter Θ_i

We compute the derivative of the cost function with respect to a any parameter $\Theta_i \neq \boldsymbol{\xi}$. The conditional densities are chosen to have the normalized exponential form $p(\boldsymbol{\xi}|j) = K(j) \cdot \exp f(\boldsymbol{\xi}, \boldsymbol{w}_j, \sigma_j^2, \Omega_j)$. Note that the normalization factor $K(j)$ depends on the shape of component j. If a mixture of n-dimensional Gaussian distributions is assumed, $K(j) = (2\pi\sigma_j^2)^{(-n/2)}$ is only valid under the constraint $\det(\Lambda_j) = 1$. We point out that the following derivatives subject to the condition $\det(\Lambda_j) = const.\ \forall j$. With $\det(\Lambda_j) = const.\ \forall j$, the $K(j)$ as defined above are scaled by a constant factor which can be disregarded. The condition of equal determinant for all j naturally includes the adaptation of a global relevance matrix $\Lambda = \Lambda_j, \forall j$.

$$
\begin{aligned}
&\frac{\partial}{\partial \Theta_i}\left(\log \frac{p(\boldsymbol{\xi}, y|\mathbf{W})}{p(\boldsymbol{\xi}|\mathbf{W})}\right) \\
&\quad = \frac{\partial \log p(\boldsymbol{\xi}, y|\mathbf{W})}{\partial \Theta_i} - \frac{\partial \log p(\boldsymbol{\xi}|\mathbf{W})}{\partial \Theta_i} \\
&\quad = \frac{1}{p(\boldsymbol{\xi}, y|\mathbf{W})} \underbrace{\frac{\partial p(\boldsymbol{\xi}, y|\mathbf{W})}{\partial \Theta_i}}_{(a)} - \frac{1}{p(\boldsymbol{\xi}|\mathbf{W})} \underbrace{\frac{\partial p(\boldsymbol{\xi}|\mathbf{W})}{\partial \Theta_i}}_{(b)} \\
&\quad = \delta_{y,c(\boldsymbol{w}_i)} \left(\frac{P(i) \exp f(\boldsymbol{\xi}, \boldsymbol{w}_i, \sigma_i^2, \Omega_i)}{p(\boldsymbol{\xi}, y|\mathbf{W})} \frac{\partial K(i)}{\partial \Theta_i} \right. \\
&\qquad \left. + \frac{P(i) K(i) \exp f(\boldsymbol{\xi}, \boldsymbol{w}_i, \sigma_i^2, \Omega_i)}{p(\boldsymbol{\xi}, y|\mathbf{W})} \frac{\partial f(\boldsymbol{\xi}, \boldsymbol{w}_i, \sigma_i^2, \Omega_i)}{\partial \Theta_i} \right) \\
&\qquad - \left(\frac{P(i) \exp f(\boldsymbol{\xi}, \boldsymbol{w}_i, \sigma_i^2, \Omega_i)}{p(\boldsymbol{\xi}|\mathbf{W})} \frac{\partial K(i)}{\partial \Theta_i} \right. \\
&\qquad \left. + \frac{P(i) K(i) \exp f(\boldsymbol{\xi}, \boldsymbol{w}_i, \sigma_i^2, \Omega_i)}{p(\boldsymbol{\xi}|\mathbf{W})} \frac{\partial f(\boldsymbol{\xi}, \boldsymbol{w}_i, \sigma_i^2, \Omega_i)}{\partial \Theta_i} \right) \\
&\quad = \delta_{y,c(\boldsymbol{w}_i)} \left(P_y(i|\boldsymbol{\xi}) - P(i|\boldsymbol{\xi}) \right) \left(\frac{1}{K(i)} \frac{\partial K(i)}{\partial \Theta_i} + \frac{\partial f(\boldsymbol{\xi}, \boldsymbol{w}_i, \sigma_i^2, \Omega_i)}{\partial \Theta_i} \right) \\
&\qquad - \left(1 - \delta_{y,c(\boldsymbol{w}_i)}\right) P(i|\boldsymbol{\xi}) \left(\frac{1}{K(i)} \frac{\partial K(i)}{\partial \Theta_i} + \frac{\partial f(\boldsymbol{\xi}, \boldsymbol{w}_i, \sigma_i^2, \Omega_i)}{\partial \Theta_i} \right)
\end{aligned}
\tag{3.27}
$$

with (a)

$$\begin{aligned}\frac{\partial p(\boldsymbol{\xi}, y|\mathbf{W})}{\partial \Theta_i} &= \frac{\partial}{\partial \Theta_i}\Big(\sum_{j:c(\boldsymbol{w}_j)=y} P(j)\, p(\boldsymbol{\xi}|j)\Big) \\ &= \sum_j \delta_{y,c(\boldsymbol{w}_j)}\, P(j)\, \frac{\partial p(\boldsymbol{\xi}|j)}{\partial \Theta_i} \\ &= \sum_j \delta_{y,c(\boldsymbol{w}_j)}\, P(j) \exp f(\boldsymbol{\xi}, \boldsymbol{w}_j, \sigma_j^2, \Omega_j) \\ &\quad \times \left(\frac{\partial K(j)}{\partial \Theta_i} + K(j)\frac{\partial f(\boldsymbol{\xi}, \boldsymbol{w}_j, \sigma_j^2, \Omega_j)}{\partial \Theta_i}\right)\end{aligned} \tag{3.28}$$

and (b)

$$\begin{aligned}\frac{\partial p(\boldsymbol{\xi}|\mathbf{W})}{\partial \Theta_i} &= \frac{\partial}{\partial \Theta_i}\left(\sum_j P(j)\, p(\boldsymbol{\xi}|j)\right) \\ &= P(i)\, \frac{\partial p(\boldsymbol{\xi}|i)}{\partial \Theta_i} \\ &= P(i)\, \exp f(\boldsymbol{\xi}, \boldsymbol{w}_i, \sigma_i^2, \Omega_i) \\ &\quad \times \left(\frac{\partial K(i)}{\partial \Theta_i} + K(i)\frac{\partial f(\boldsymbol{\xi}, \boldsymbol{w}_i, \sigma_i^2, \Omega_i)}{\partial \Theta_i}\right)\end{aligned} \tag{3.29}$$

$P_y(i|\boldsymbol{\xi})$ corresponds to the probability that sample $\boldsymbol{\xi}$ is assigned to component i of the correct class y and $P(i|\boldsymbol{\xi})$ depicts the probability the $\boldsymbol{\xi}$ is assigned to any component i of the mixture.

$$\begin{aligned}P_y(i|\boldsymbol{\xi}) &= \frac{P(i)K(i) \exp f(\boldsymbol{\xi}, \boldsymbol{w}_i, \sigma_i^2, \Omega_i)}{p(\boldsymbol{\xi}, y|\mathbf{W})} \\ &= \frac{P(i)K(i) \exp f(\boldsymbol{\xi}, \boldsymbol{w}_i, \sigma_i^2, \Omega_i)}{\sum_{j:c(\boldsymbol{w}_j)=y} P(j)K(j) \exp f(\boldsymbol{\xi}, \boldsymbol{w}_j, \sigma_j^2, \Omega_j)},\end{aligned} \tag{3.30}$$

$$\begin{aligned}P(i|\boldsymbol{\xi}) &= \frac{P(i)K(i) \exp f(\boldsymbol{\xi}, \boldsymbol{w}_i, \sigma_i^2, \Omega_i)}{p(\boldsymbol{\xi}|\mathbf{W})} \\ &= \frac{P(i)K(i) \exp f(\boldsymbol{\xi}, \boldsymbol{w}_i, \sigma_i^2, \Omega_i)}{\sum_j P(j)K(j) \exp f(\boldsymbol{\xi}, \boldsymbol{w}_j, \sigma_j^2, \Omega_j)}\end{aligned} \tag{3.31}$$

The derivative with respect to a global parameter, e.g. a global matrix $\Omega = \Omega_j$ for all j can be derived thereof by summation.

Material based on:

Petra Schneider, Kerstin Bunte, Han Stiekema, Barbara Hammer, Thomas Villmann and Michael Biehl - *"Regularization in Matrix Relevance Learning,"* IEEE Transactions on Neural Networks, vol. 21, no. 5, 2010.

Chapter 4

Regularization in matrix learning

Abstract

We present a regularization technique to extend recently proposed matrix learning schemes in Learning Vector Quantization (LVQ). These learning algorithms extend the concept of adaptive distance measures in LVQ to the use of relevance matrices. In general, metric learning can display a tendency towards over-simplification in the course of training. An overly pronounced elimination of dimensions in feature space can have negative effects on the performance and may lead to instabilities in the training. We focus on matrix learning in Generalized LVQ. Extending the cost function by an appropriate regularization term prevents the unfavorable behavior and can help to improve the generalization ability. The approach is first tested and illustrated in terms of artificial model data. Furthermore, we apply the scheme to a benchmark classification data set from the UCI Repository of machine learning. We demonstrate the usefulness of regularization also in the case of rank limited relevance matrices, i.e. matrix learning with an implicit, low dimensional representation of the data.

4.1 Introduction

Metric learning is a valuable technique to improve the basic LVQ approach of nearest prototype classification: a parameterized distance measure is adapted to the data to optimize the metric for the specific application. Relevance learning allows to weight the input features according to their importance for the classification task (Bojer et al., 2001; Hammer and Villmann, 2002). Especially in case of high dimensional, heterogeneous real life data this approach turned out particularly suitable, since it accounts for irrelevant or inadequately scaled dimensions; see Mendenhall and Merényi (2006); Kietzmann et al. (2008) for applications. Matrix learning additionally accounts for pairwise correlations of features (Schneider et al., 2009a,b); hence, very flexible distance measures can be derived.

However, metric adaptation techniques may be subject to over-simplification of the classifier as the algorithms possibly eliminate too many dimensions. A theoretical investigation for this behavior is derived in Biehl, Hammer, Schleif, Schneider and Villmann (2009).

In this work, we present a regularization scheme for metric adaptation methods in LVQ to prevent the algorithms from over-simplifying the distance measure. We demonstrate the behavior of the method by means of an artificial data set and a real world application. It is also applied in the context of rank limited relevance matrices which realize an implicit low-dimensional representation of the data.

4.2 Motivation

The standard motivation for regularization is to prevent a learning system from over-fitting, i.e. the overly specific adaptation to the given training set. In previous applications of matrix learning in LVQ we observe only weak over-fitting effects. Nevertheless, restricting the adaptation of relevance matrices can help to improve generalization ability in some cases.

A more important reasoning behind the suggested regularization is the following: in previous experiments with different metric adaptation schemes in Learning Vector Quantization it has been observed that the algorithms show a tendency to over-simplify the classifier (Biehl, Breitling and Li, 2007; Schneider et al., 2009a), i.e. the computation of the distance values is finally based on a strongly reduced number of features compared to the original input dimensionality of the data. In case of matrix learning in LVQ1, this convergence behavior can be derived analytically under simplifying assumptions (Biehl, Hammer, Schleif, Schneider and Villmann, 2009). The elaboration of these considerations is ongoing work and will be topic of further forthcoming publications. Certainly, the observations described above indicate that the arguments are still valid under more general conditions. Frequently, there is only one linear combination of features remaining at the end of training. Depending on the adaptation of a relevance vector or a relevance matrix, this results in a single non-zero relevance factor or eigenvalue, respectively. Observing the evolution of the relevances or eigenvalues in such a situation shows that the classification error either remains constant while the metric still adapts to the data, or the over-simplification causes a degrading classification performance on training and test set. Note that these observations do not reflect over-fitting, since training and test error increase concurrently. In case of the cost-function based algorithms this effect could be explained by the fact that a minimum of the cost function does not necessarily coincide with an optimum in matters of classification performance.

Note that the numerator in Eq. (3.10) is smaller than 0 iff the classification of the data point is correct. The smaller the numerator, the greater the security of classification, i.e. the difference of the distances to the closest correct and wrong prototype. While this effect is desirable to achieve a large separation margin, it has unwanted effects when combined with metric adaptation: it causes the risk of a complete deletion of dimensions if they contribute only minor parts to the classification. This way, the classification accuracy might be severely reduced in exchange for sparse, 'over-simplified' models. But over-simplification is also observed in training with heuristic algorithms (Biehl, Breitling and Li, 2007). Training of relevance vectors seems to be more sensitive to this effect than matrix adaptation. The determination of a new direction in feature space allows more freedom than the restriction to one of the original input features. Nevertheless, degrading classification performance can also be expected for matrix adaptation. Thus, it may be reasonable to improve the learning behavior of matrix adaptation algorithms by preventing strong decays in the eigenvalue profile of Λ.

In addition, extreme eigenvalue settings can invoke numerical instabilities in case of GMLVQ. An example scenario, which involves an artificial data set, will be presented in the Sec. 4.5.1. Our regularization scheme prevents the matrix Λ from becoming singular. As we will demonstrate, it thus overcomes the above mentioned instability problem.

4.3 General approach

In order to derive relevance matrices with less distinct eigenvalue profiles, we make use of the fact that maximizing the determinant of an arbitrary, quadratic matrix $A \in \mathbb{R}^{n\times n}$ with eigenvalues $\nu_1, \ldots, \nu_n$ suppresses large differences between the ν_i. Note that $\det(A) = \prod_i \nu_i$ which is maximized by $\nu_i = 1/n, \forall i$ under the constraint $\sum_i \nu_i = 1$. Hence, maximizing $\det(\Lambda)$ seems to be an appropriate strategy to manipulate the eigenvalues of Λ the desired way, if Λ is non-singular. However, since $\det(\Lambda) = 0$ holds for $\Omega \in \mathbb{R}^{m\times n}$ with $m < n$, this approach cannot be applied, if the computation of Λ is based on a rectangular matrix Ω. But note that the first m eigenvalues of $\Lambda = \Omega^\top\Omega$ are equal to the eigenvalues of $\Omega\Omega^\top \in \mathbb{R}^{m\times m}$. Hence, maximizing $\det(\Omega\Omega^\top)$ imposes a tendency of the first m eigenvalues of Λ to reach the value $1/m$. Since $\det(\Lambda) = \det(\Omega^\top\Omega) = \det(\Omega\Omega^\top)$ holds for $m = n$, we propose the following regularization term θ in order to obtain a relevance matrix Λ with balanced eigenvalues close to $1/n$ or $1/m$ respectively:

$$\theta = \ln\left(\det\left(\Omega\Omega^\top\right)\right). \tag{4.1}$$

The approach can easily be applied to any LVQ algorithm with an underlying

cost function E. Note that θ has to be added or subtracted depending on the character of E. The derivative with respect to Ω yields

$$\begin{aligned}\frac{\partial\theta}{\partial\Omega} &= \frac{\partial \ln(\det(\Omega\Omega^\top))}{\partial \det(\Omega\Omega^\top)} \frac{\partial \det(\Omega\Omega^\top)}{\partial\Omega\Omega^\top} \frac{\partial\Omega\Omega^\top}{\partial\Omega} \\ &= 2 \cdot (\Omega^+)^\top,\end{aligned} \tag{4.2}$$

where Ω^+ denotes the Moore-Penrose pseudo-inverse of Ω. For the proof of this derivative we refer to Petersen and Pedersen (2008).

The concept can easily be transfered to relevance LVQ with exclusively diagonal relevance factors (Bojer et al., 2001; Hammer and Villmann, 2002): in this case the regularization term reads $\theta = \ln(\prod_i \lambda_i)$, since the weight factors λ_i in the scaled Euclidean metric correspond to the eigenvalues of Λ. Since θ is only defined in terms of the metric parameters, it can be expected that the number of prototypes does not have significant influence on the application of the regularization technique.

4.4 Learning rules for GMLVQ

In the experimental section, we apply the proposed regularization technique to different algorithms based on the GLVQ cost function. The extended cost function of GMLVQ (see Eq. (3.10)) yields

$$\tilde{E}_{\mathrm{GMLVQ}} = E_{\mathrm{GMLVQ}} - \frac{\eta}{2} \cdot \ln\left(\det\left(\Omega\Omega^\top\right)\right), \tag{4.3}$$

where the regularization parameter η adjusts the importance of the different goals covered by $\tilde{E}_{\mathrm{GMLVQ}}$. The parameter has to be optimized by means of a validation procedure. Since the regularization term does not include the prototype positions, the learning rules for $\boldsymbol{w}_J$ and $\boldsymbol{w}_K$ as specified in Eq.s (3.11) and (3.12) do not change due to the regularization. The new update for the metric parameter yields

$$\Delta\Omega_{lm} = -\alpha_2 \cdot \frac{\partial E_{\mathrm{GMLVQ}}}{\partial\Omega_{lm}} + \alpha_2 \cdot \eta \cdot \Omega^+_{ml}, \tag{4.4}$$

where the first summand is given in Eq. (3.13). In our experiments, we also apply the proposed regularization approach to GRLVQ and Local GMLVQ.

4.5 Experiments

In the following experiments, we always initialize the relevance matrix Λ with the identity matrix followed by a normalization step such that $\sum_i \Lambda_{ii} = 1$. As initial prototypes, we choose the mean values of random subsets of training samples selected from each class.

4.5.1 Artificial data

The first illustrative application is the artificial data set visualized in Fig. 4.1. It constitutes a binary classification problem in a two-dimensional space. Training and validation data are generated according to axis-aligned Gaussians of 600 samples with mean $\mu_1 = [1.5, 0.0]$ for class 1 and $\mu_2 = [-1.5, 0.0]$ for class 2 data, respectively. In both classes the standard deviations are $\sigma_{11} = 0.5$ and $\sigma_{22} = 3.0$. These clusters are rotated independently by the angles $\varphi_1 = \pi/4$ and $\varphi_2 = -\pi/6$ so that the two clusters intersect. To verify the results, we perform the experiments on ten independently generated data sets.

At first, we focus on the adaptation of a global relevance matrix by GMLVQ. We use the learning rates $\alpha_1 = 0.01$ and $\alpha_2 = 1 \cdot 10^{-3}$ and train the system for 100 epochs. In all experiments, the behavior described in (Biehl, Hammer, Schleif, Schneider and Villmann, 2009) is visible immediately; Λ reaches the eigenvalue settings one and zero within 10 sweeps through the training set. Hence, the system uses a one-dimensional subspace to discriminate the data. This subspace stands out due to minimal data variance around the respective prototype of one class. Accordingly, this subspace is defined by the eigenvector corresponding to the smallest eigenvalue of the class specific covariance matrix. This issue is illustrated in Fig.s 4.1 (a) and (d). Due to the nature of the data set, this behavior leads to a very poor representation of the samples belonging to the other class by the respective prototype which implies a very weak class-specific classification performance as depicted by the receptive fields.

However, numerical instabilities can be observed, if local relevance matrices are trained for this data set. In accordance with the theory in (Biehl, Hammer, Schleif, Schneider and Villmann, 2009), the matrices become singular in only a small number of iterations. Projecting the samples onto the second eigenvector of the class specific covariance matrices allows to realize minimal data variance around the respective prototype for both classes (see Fig.s 4.1 (e), (f)). Consequently, the great majority of data points obtains very small values d_J^Λ and comparably large values d_K^Λ. But samples lying in the overlapping region yield very small values for both distances d_J^Λ and d_K^Λ. In consequence, they cause abrupt, large parameter updates for the prototypes and the matrix elements (see Eq.s (3.11), (3.12), (3.14)). This leads to instable training behavior and peaks in the learning curve as can be seen in Fig. 4.2.

Applying the proposed regularization technique leads to a much smoother learning behavior. With $\eta = 0.005$, the matrices $\Lambda_{1,2}$ do not become singular and the peaks in the learning curve are eliminated (see Fig. 4.2). Misclassifications only occur in case of data lying in the overlapping region of the clusters; the system

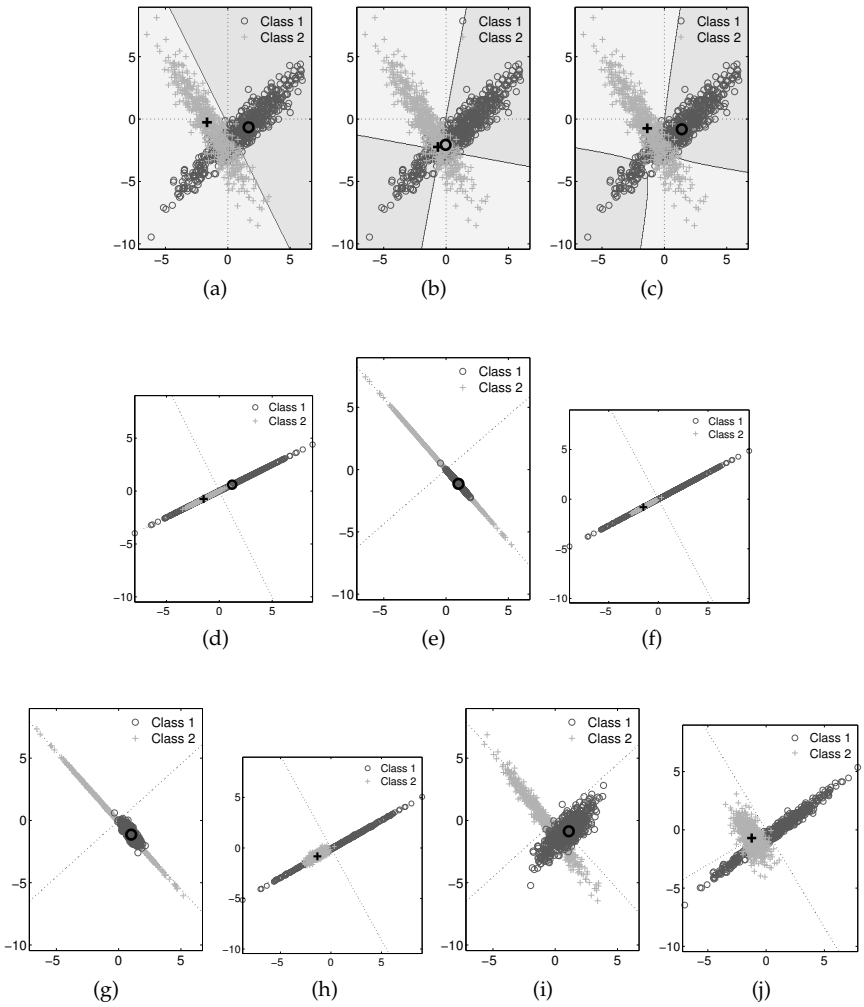


Figure 4.1: Artificial data. *(a) - (c) Prototypes and receptive fields, (a) GMLVQ without regularization, (b) LGMLVQ without regularization, (c) LGMLVQ with $\eta = 0.15$, (d) Training set transformed by global matrix Ω after GMLVQ training, (e), (f) Training set transformed by local matrices Ω_1, Ω_2 after LGMLVQ training, (g), (h) Training set transformed by local matrices Ω_1, Ω_2 obtained by LGMLVQ Training with $\eta = 0.005$, (i), (j) Training set transformed by local matrices Ω_1, Ω_2 obtained by LGMLVQ Training with $\eta = 0.15$. In (d) - (j) the dotted lines correspond to the eigendirections of Λ or Λ_1 and Λ_2, respectively.*

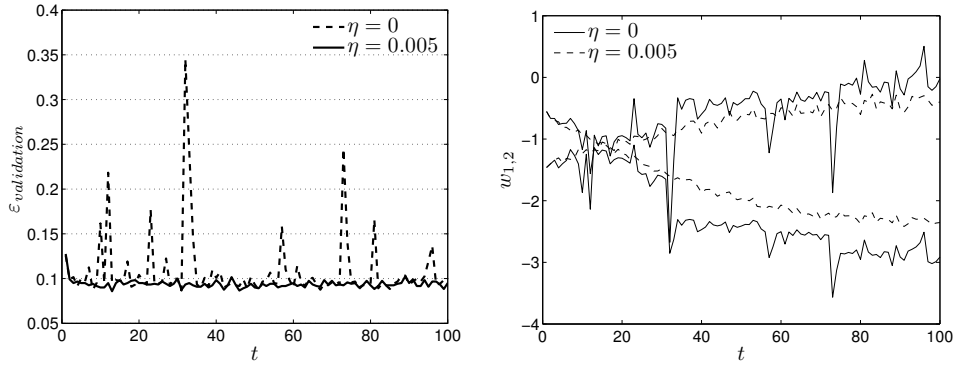


Figure 4.2: Artificial data. *The plots relate to experiments on a single data set.* **Left:** *Evolution of error rate on validation set during LGMLVQ-Training with* $\eta = 0$ *and* $\eta = 0.005$. **Right:** *Coordinates of the class 2 prototype during LGMLVQ-Training with* $\eta = 0$ *and* $\eta = 0.005$.

achieves $\varepsilon_{validation} = 9\%$. The relevance matrices exhibit the mean eigenvalues $\text{eig}(\Lambda_{1,2}) \approx (0.99, 0.01)$. Accordingly, the samples spread slightly in two dimensions after transformation with Ω_1 and Ω_2 (see Fig.s 4.1 (g), (h)). An increasing number of misclassifications can be observed for $\eta > 0.1$. Fig.s 4.1 (c), (i), (j) visualize the results of running LGMLVQ with the new cost function and $\eta = 0.15$. The mean eigenvalue profiles of the relevance matrices obtained in these experiments are $\text{eig}(\Lambda_1) \approx (0.83, 0.17)$ and $\text{eig}(\Lambda_2) \approx (0.84, 0.16)$. The mean test error at the end of training saturates at $\varepsilon_{validation} \approx 13\%$.

4.5.2 Real life data

In the second set of experiments, we apply the algorithms to two benchmark data set provided by the UCI-Repository of Machine Learning (Newman et al., 1998), namely Pima Indians Diabetes and Letter Recognition.

Pima Indians Diabetes

The data set constitutes a binary classification problem in an eight dimensional feature space. It has to be predicted, whether an at least 21 years old female of Pima Indian heritage shows signs of diabetes according to the World Health Organization criteria. The data set contains 768 instances, 500 class 1 samples (diabetes) and 268 class 2 samples (healthy). As a preprocessing step, a z-transformation is applied to normalize all features to zero mean and unit variance.

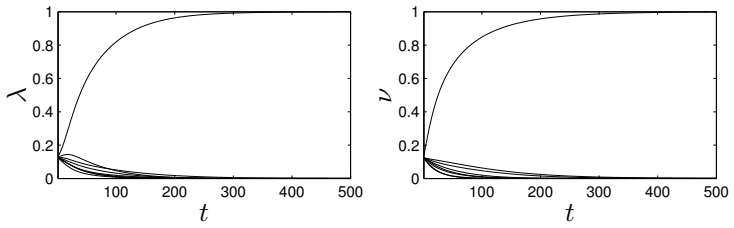


Figure 4.3: Pima indians diabetes data. *Evolution of relevance values* $\boldsymbol{\lambda}$ *(left) and eigenvalues* $\boldsymbol{\nu} = eig(\Lambda)$ *(right) during training of GRLVQ and GMLVQ with* $\Omega \in \mathbb{R}^{8\times 8}$ *observe in one training run.*

We split the data set randomly into 2/3 for training and 1/3 for validation and average the results over 30 such random splits. We approximate the data by means of one prototype per class. The learning rates are chosen as follows: $\alpha_1 = 1 \cdot 10^{-3}$, $\alpha_2 = 1 \cdot 10^{-4}$. The regularization parameter is chosen from the interval [0, 1.0]. We use the weighted Euclidean metric (GRLVQ) and GMLVQ with $\Omega \in \mathbb{R}^{8\times 8}$ and $\Omega \in \mathbb{R}^{2\times 8}$. The system is trained for 500 epochs in total.

Using the standard GLVQ cost function without regularization, we observe that the metric adaptation with GRLVQ and GMLVQ leads to an immediate selection of a single feature to classify the data. Fig. 4.3 visualizes examples of the evolution of relevances and eigenvalues in the course of relevance and matrix learning based on one specific training set. GRLVQ bases the classification on feature 2: Plasma glucose concentration, which is also a plausible result from the medical point of view.

Fig. 4.4a illustrates how the regularization parameter η influences the performance of GRLVQ. Using small values of η reduces the mean rate of misclassification on training and validation sets compared to the non-regularized cost function. We observe the optimum classification performance on the validation sets for $\eta \approx 0.03$; the mean error rate constitutes $\varepsilon_{validation} = 25.2\%$. However, the range of regularization parameters which achieve a comparable performance is quite small. The classifiers obtained with $\eta > 0.06$ already perform worse compared to the original GRLVQ-algorithm. Hence, the system is very sensitive with respect to the parameter η.

Next, we discuss the GMLVQ results obtained with $\Omega \in \mathbb{R}^{8\times 8}$. As depicted in Fig. 4.4b, restricting the algorithm with the proposed regularization method improves the classification of the validation data slightly; the mean performance on the validation sets increases for small values of η and reaches $\varepsilon_{validation} \approx 23.4\%$ with $\eta = 0.015$. The improvement is weaker compared to GRLVQ, but note that the decreasing validation error is accompanied by an increasing training error. Hence,

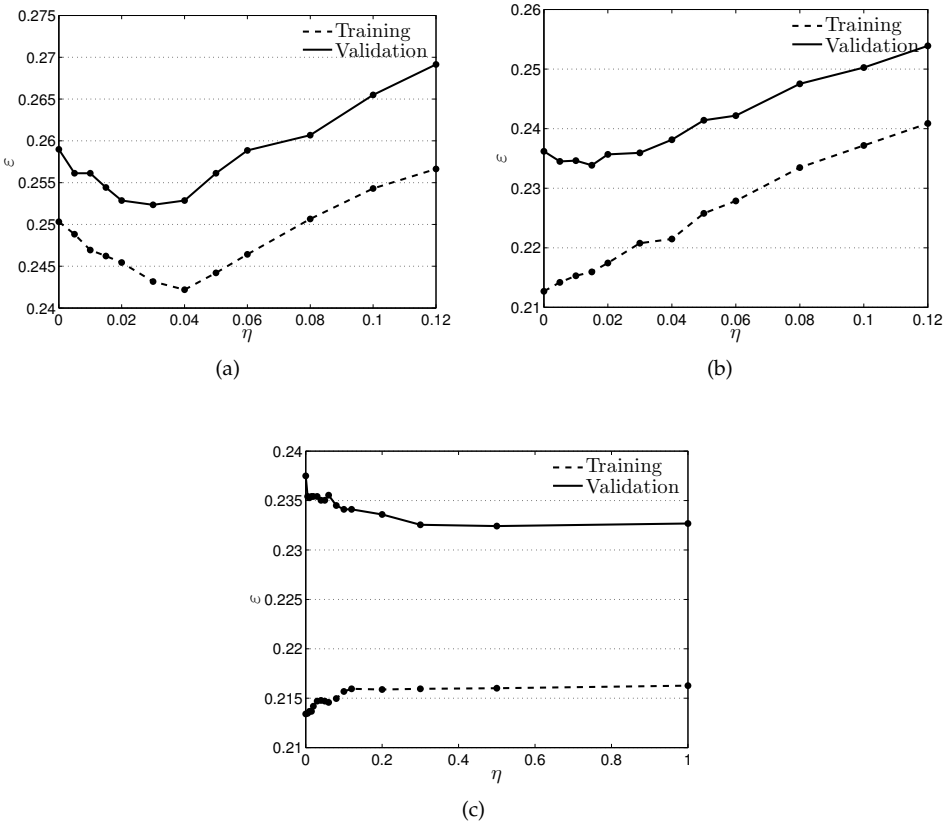


Figure 4.4: Pima indians diabetes data. *Mean error rates on training and validation sets after training different algorithms with different regularization parameters η. (a) GRLVQ. (b) GMLVQ with $\Omega \in \mathbb{R}^{8\times 8}$. (c) GMLVQ with $\Omega \in \mathbb{R}^{2\times 8}$.*

the specificity of the classifier with respect to the training data is reduced; the regularization helps to prevent over-fitting. Note that this over-fitting effect could not be overcome by an early stopping of the unrestricted learning procedure.

Similar observations can be made for GMLVQ with $\Omega \in \mathbb{R}^{2\times 8}$. The regularization slightly improves the performance on the validation set while the accuracy on the training data is degrading (see Fig. 4.4c). Since the penalty term in the cost function becomes much larger for matrix adaptation with $\Omega \in \mathbb{R}^{2\times 8}$, larger values for η are necessary in order to reach the desired effect on the eigenvalues of $\Omega\Omega^\top$. The plot in Fig. 4.4 depicts that the error on the validation sets reaches a stable optimum for $\eta > 0.3$; $\varepsilon_{validation} = 23.3\%$. The increasing validation set performance is also

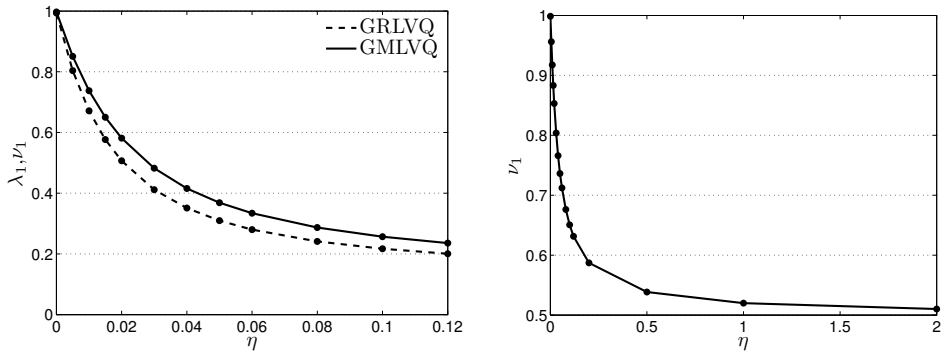


Figure 4.5: Pima indians diabetes data. *Dependency of the largest relevance value λ_1 in GRLVQ and the largest eigenvalue ν_1 in GMLVQ on the regularization parameter η. The plots are based on the mean relevance factors and mean eigenvalues obtained with the different training sets at the end of training.* **Left:** *Comparison between GRLVQ and GMLVQ with $\Omega \in \mathbb{R}^{8\times 8}$.* **Right:** *GMLVQ with $\Omega \in \mathbb{R}^{2\times 8}$.*

accompanied by a decreasing performance on the training sets.

Fig. 4.5 visualizes how the values of the largest relevance factor and the first eigenvalue depend on the regularization parameter. With increasing η, the values converge to $1/n$ or $1/m$, respectively. Remarkably, the curves are very smooth.

The coordinate transformation defined by $\Omega \in \mathbb{R}^{2\times 8}$ allows to construct a two-dimensional representation of the data set which is particularly suitable for visualization purposes. In the low-dimensional space, the samples are scaled along the coordinate axes according to the features' relevance for classification. Due to the fact that the relevances are given by the eigenvalues of $\Omega\Omega^\top$, the regularization technique allows to obtain visualizations which separate the classes more clearly. This effect is illustrated in Fig. 4.6 which visualizes the prototypes and the data after transformation with one matrix Ω obtained in a single training run. Due to the over-simplification with $\eta = 0$ the samples are projected onto a one-dimensional subspace. Visual inspection of this representation does not provide further insight into the nature of the data. On the contrary, after training with $\eta = 2.0$ the data is almost equally scaled in both dimensions, resulting in a discriminative visualization of the two classes.

SVM results reported in the literature can be found e.g. in Ong et al. (2005); Tamura and Tanno (2008). The error rates on test data vary between 19.3% and 27.2%. However, we would like to stress that our main interest in the experiments is related to the analysis of the regularization approach in comparison to original

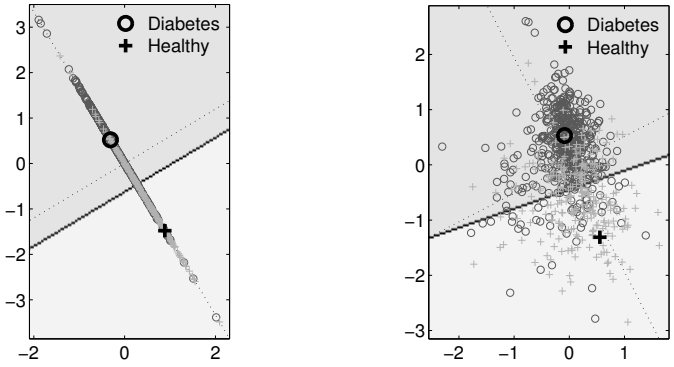


Figure 4.6: Pima indians diabetes data. *Two-dimensional representation of the complete data set found by GMLVQ with* $\Omega \in \mathbb{R}^{2\times 8}$ *and* $\eta = 0$ *(left),* $\eta = 2.0$ *(right) obtained in one training run. The dotted lines correspond to the eigendirections of* $\Omega\Omega^\top$.

GMLVQ. For this reason, further validation procedures to optimize the classifiers are not examined in this study.

Letter Recognition

The data set consists of 20 000 feature vectors encoding different attributes of black-and-white pixel displays of the 26 capital letters of the English alphabet. We split the data randomly in a training and a validation set of equal size and average our results over 10 independent random compositions of training and validation set. First, we adapt one prototype per class. We use $\alpha_1 = 1 \cdot 10^{-3}$, $\alpha_2 = 1 \cdot 10^{-4}$ and test regularization parameters from the interval [0, 0.1]. The dependence of the classification performance on the value of the regularization parameter for our GRLVQ and GMLVQ experiments are depicted in Fig. (4.7). It is clearly visible that the regularization improves the performance for small values of η compared to the experiments with $\eta = 0$. Compared to global GMLVQ, the adaptation of local relevance matrices improves the classification accuracy significantly; we obtain $\varepsilon_{validation} \approx 12\%$. Since no over-fitting or over-simplification effects are present in this application, the regularization does not achieve further improvements anymore.

Additionally, we perform GMLVQ training with three prototypes per class. The learning rates are set slightly larger to $\alpha_1 = 5 \cdot 10^{-3}$ and $\alpha_2 = 5 \cdot 10^{-4}$ in order to increase the speed of convergence; the system is trained for 500 epochs. We observe that the algorithm's behavior resembles the previous experiments with only one prototype per class. Already small values η effect a significant reduction of

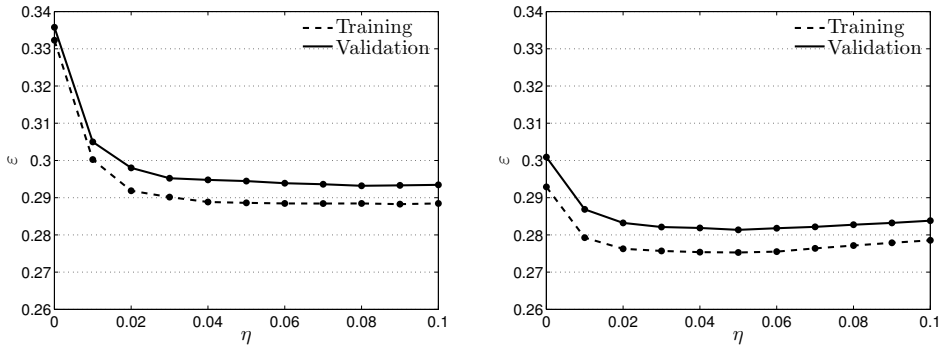


Figure 4.7: Letter recognition data set *Mean error rates on training and validation sets after training different algorithms with different regularization parameters η.* **Left:** *GRLVQ.* **Right:** *GMLVQ with* $\Omega \in \mathbb{R}^{16\times 16}$.

the mean rate of misclassification. Here, the optimal value η is the same for both model settings. With $\eta = 0.05$, the classification performance improves about $\approx 2\%$ compared to training with $\eta = 0$ (see Fig.s 4.8, left). Furthermore, the shape of the eigenvalue profile of Λ is nearly independent of the codebook size as depicted Fig. 4.8, right. These observations support the statement that the regularization and the number of prototypes can be varied independently.

4.6 Conclusion

We proposed a regularization technique to extend matrix learning schemes in Learning Vector Quantization. The study is motivated by the behavior analysed in Biehl, Hammer, Schleif, Schneider and Villmann (2009): matrix learning tends to perform an overly strong feature selection which may have negative impact on the classification performance and the learning dynamics. We introduce a regularization scheme which inhibits strong decays in the eigenvalue profile of the relevance matrix. The method is very flexible: it can be used in combination with any cost function and is also applicable to the adaptation of relevance vectors.

Here, we focus on matrix adaptation in Generalized LVQ. The experimental findings highlight the practicability of the proposed regularization term. It is shown in artificial and a real life application that the technique tones down the algorithm's feature selection. In consequence, the proposed regularization scheme prevents over-simplification, eliminates instabilities in the learning dynamics and improves the generalization ability of the considered metric adaptation algorithms.

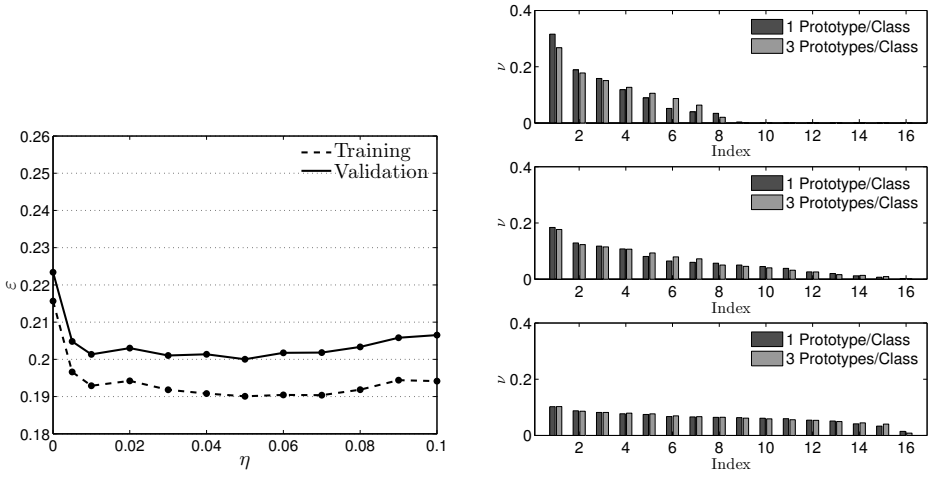


Figure 4.8: Letter recognition data set **Left:** *Mean error rates on training and validation set after GMLVQ training with* $\Omega \in \mathbb{R}^{16\times16}$ *and three prototypes per class with different regularization parameters* η *.* **Right** *Comparison of mean eigenvalue profiles of final matrix* Λ *obtained by GMLVQ training (*$\Omega \in \mathbb{R}^{16\times16}$*) with different numbers of prototypes and different regularization parameters. Top:* $\eta = 0$*. Middle:* $\eta = 0.01$*. Bottom:* $\eta = 0.05$*.*

Beyond, our method turns out to be advantageous to derive discriminative visualizations by means of GMLVQ with a rectangular matrix Ω.

However, these effects highly depend on the choice of an appropriate regularization parameter η which has to be determined by means of a validation procedure. A further drawback constitutes the matrix inversion included in the new learning rules since it is computationally expensive operation. Future projects will concern the application of the regularization method in very high-dimensional data. There, the computational costs of the matrix inversion which is required in the relevance updates can become problematic. However, efficient techniques for the iteration of an approximate pseudo-inverse can be developed which make the method also applicable to classification problems in high dimensional spaces.

Material based on:

Michael Biehl and Petra Schneider and Barbara Hammer and Frank-Michael Schleif and Thomas Villmann- *"Stationarity of Matrix Updates in Relevance Learning Vector Quantization,"* submitted, 2010.

Chapter 5

Stationarity of matrix learning

Abstract

We investigate the convergence properties of matrix learning, a framework for the use of adaptive distance measures in Learning Vector Quantization. Under simplifying assumptions on the training process, stationarity conditions can be worked out which characterize the outcome of training in terms of the relevance matrix. We show that, generically, the training process singles out one specific direction or a low-dimensional subspace in feature space which depends on the statistical properties of the data and the approached prototype configuration. We also work out the stationarity conditions for regularized matrix learning which can be used to favor full rank relevance matrices in order to prevent over-simplification effects. The structure of the stationary solution is derived, giving insight into the influence of the regularization parameter. In addition, illustrative computer experiments are presented.

5.1 Introduction

Similarity based methods play an important role in supervised and unsupervised machine learning tasks, such as classification, regression or clustering. For a recent overview see, for instance, (Biehl, Hammer, Verleysen and Villmann, 2009). Most methods employ pre-defined measures to evaluate a generalized distance between vectors in feature space. The by far most popular choices are the standard Minkowski measures and generalizations thereof such as weighted Euclidean or Manhattan distance.

The key problem is, of course, the identification of a suitable distance measure which is appropriate for the problem at hand. Available insights into the scaling of different features or relations between them can be taken advantage of in the construction of a specific weighting scheme, for instance. However, frequently, a priori knowledge is limited and it is unclear which choice would facilitate good performance.

A particular attractive concept are adaptive distance measures which change in the course of learning. In this report, we discuss adaptive metrics in the context of Learning Vector Quantization (LVQ, Kohonen, 1997). This family of supervised, distance based classification schemes has proven useful in a variety of practical problems and adaptive metrics can incorporate in a straightforward way. Here, we consider adaptive distance measures in LVQ systems for $n-$dimensional data which are parameterized by an $n \times n$ relevance matrix (Schneider et al., 2007, 2009a,b). This generalization of the Euclidean distance allows for the weighting of single features and, in addition, takes into account pairwise correlations between the features through off-diagonal matrix entries.

While metric adaptation has proven useful in many practical applications, a better theoretical understanding is desirable. Essential questions concern, for instance, the convergence behavior of the learning algorithms and the uniqueness of the obtained solution. In the following sections, we investigate properties of stationary relevance matrices in metric learning. We show that generic update rules display a tendency to yield a low rank measure which takes into account only a single or very few directions in feature space. On the one hand, this effect can help to avoid overfitting effects in practical situations since it limits the complexity of the distance measure. On the other hand, this tendency can lead to deteriorating performance due to over-simplification and to numerical instabilities as matrices become singular. Regularization methods can be applied to cure these problems, we consider a particular strategy which allows to control the complexity of the relevant eigenspectra continuously (Schneider et al., 2010).

5.2 Learning algorithm

We study the convergence of matrix learning in the framework of LVQ1 and its extension by local relevance matrices. The following summarizes a heuristic extension of LVQ1 along the lines of local relevance matrix learning for a set of prototypes $\{\boldsymbol{w}_j\}_{j=1}^l \in \mathbb{R}^n$ carrying class labels $\{c(\boldsymbol{w}_j)\}_{j=1}^l \in \{1,\ldots,C\}$. Note that we use a slightly different notation throughout this chapter, to be consistent with the notation in Biehl et al. (submitted, 2010). In the following, the generalized Euclidean distance $d^\Lambda(\boldsymbol{w},\boldsymbol{\xi})$ (see Eq. (3.1)) is defined in terms of $\Lambda = \Gamma\,\Gamma^\top$, where $\Gamma \in \mathbb{R}^{n\times n}$. No constraints are imposed on the symmetry or structure of Γ. Note that Γ corresponds to $\Omega^\top$ as introduced in Sec. 3.2. Hence, the employed distance measure reads

$$d^\Lambda(\boldsymbol{w},\boldsymbol{\xi}) = (\boldsymbol{\xi}-\boldsymbol{w})\Gamma\,\Gamma^\top(\boldsymbol{\xi}-\boldsymbol{w}) = \sum_{ijk}(\xi^i - w^i)\Gamma_{ik}\Gamma_{jk}(\xi^j - w^j). \tag{5.1}$$

An update of the model parameters in local Matrix LVQ1 consists of the following steps:

1. Randomly select a training sample $(\boldsymbol{\xi}, y)$
2. Determine the winning prototype $\boldsymbol{w}_L$ with $d^{\Lambda_L}(\boldsymbol{w}_L, \boldsymbol{\xi}) = \min_l \{d^{\Lambda_l}(\boldsymbol{w}_l, \boldsymbol{\xi})\}$, breaking ties arbitrarily
3. Update $\boldsymbol{w}_L$ according to

$$\boldsymbol{w}_L \leftarrow \boldsymbol{w}_L + \alpha_1 \, \psi(c(\boldsymbol{w}_L), y) \, \Gamma_L \, \Gamma_L^\top \, (\boldsymbol{\xi} - \boldsymbol{w}_L), \tag{5.2}$$

   where $\psi(c(\boldsymbol{w}), y) = 1$, if $c(\boldsymbol{w}) = y$ and $\psi(c(\boldsymbol{w}), y) = -1$ otherwise.

4. Update the local matrix Γ_L according to

$$\begin{aligned} \Gamma_L &\leftarrow \Gamma_L - \alpha_2 \, \psi(c(\boldsymbol{w}_L), y) \, \frac{1}{2} \, \frac{\partial d(\boldsymbol{w}_L, \boldsymbol{\xi})}{\partial \Gamma_L} \\ &= \Gamma_L - \alpha_2 \, \psi(c(\boldsymbol{w}_L), y) \, \boldsymbol{x} \, \boldsymbol{x}^\top \, \Gamma_L, \end{aligned} \tag{5.3}$$

   with the abbreviation $\boldsymbol{x} = (\boldsymbol{\xi} - \boldsymbol{w}_L)$. Alternatively, if regularization is included in the learning process, the matrix update constitutes

$$\Gamma_L \leftarrow \Gamma_L - \alpha_2 \, \psi(c(\boldsymbol{w}_L), y) \, \boldsymbol{x}\boldsymbol{x}^\top \, \Gamma_L + \alpha_2 \, \eta \, \Gamma_L^{-\top}, \tag{5.4}$$

   where η is the regularization parameter (see chapter 4 for details).

5.3 Analysis of stationarity

We discuss the convergence of the above defined matrix updates under simplifying assumptions. We work out stationarity conditions on the considered learning dynamics by performing a formal average of the update over the data drawn from the given training set. For the following arguments, we make the simplifying assumption that prototypes can be considered stationary, either because we set $\alpha_1 = 0$ explicitly, or because they have converged after an initial training phase. The analysis further assumes that a fixed subset of the data is assigned to each of the prototypes. In other words: We assume that, in the limit of large training times, the assignment of example data to the prototypes does not change any more even though the distance measure is modified in the training. One scenario in which this condition should be satisfied is that of data drawn from well separated clusters which are assigned to different prototypes. In such a situation, small modifications of the metric

may slightly change the shape of receptive fields, but their borders fall into the gaps between clusters and will remain unchanged. In subsection 5.4, we point out implications of this condition for practical data sets and discuss to what extent it may be relaxed with respect to the applicability of the theory.

Assuming the assignment of inputs to prototypes is fixed, we can focus on a single prototype $\boldsymbol{w}_L = const.$, its local relevance matrix Γ_L, and the fixed set of vectors $\boldsymbol{\xi}$ assigned to $\boldsymbol{w}_L$. In the following, we omit the index that identifies the winning prototype $\boldsymbol{w}$ with label $c(\boldsymbol{w})$. It is implicitly understood that the local matrix is updated only from the corresponding subset of data. To derive averaged quantities, which take assignments of data to prototypes into account, we use the shorthand notation

$$\langle\, f \,\rangle = \sum_j f(\boldsymbol{\xi}_j)\, \phi(\boldsymbol{\xi}_j, \boldsymbol{w}) \Big/ \sum_j \phi(\boldsymbol{\xi}_j, \boldsymbol{w}), \tag{5.5}$$

where f is the quantity of interest. The sum is formally over all examples and the indicator function $\phi(\boldsymbol{\xi}, \boldsymbol{w}) = 1$, if $\boldsymbol{w}$ is the prototype closest to $\boldsymbol{\xi}$ and $\phi(\boldsymbol{\xi}, \boldsymbol{w}) = 0$ else.

5.3.1 Unregularized matrix updates

First, we analyse unrestricted matrix updates as specified in Eq. (5.3). In time step μ of this training (sub-) process, an example $(\boldsymbol{\xi}, y)$ is presented. Before normalization, the update of the corresponding matrix reads

$$\Gamma(\mu+1) \;\sim\; \Gamma(\mu) - \alpha_2\, \psi(c(\boldsymbol{w}), y)\, \boldsymbol{x}\, \boldsymbol{x}^\top\, \Gamma(\mu),$$

with $\boldsymbol{x} = (\boldsymbol{\xi} - \boldsymbol{w})$. We consider the update on average over all examples in the subset of data assigned to $\boldsymbol{w}$. Using the notation in Eq. (5.5), the averaged update results in

$$\Gamma \propto \Gamma - \alpha_2 \left\langle \psi(c(\boldsymbol{w}), y)\, \boldsymbol{x}\boldsymbol{x}^\top \right\rangle \Gamma \equiv [\mathbb{1} - \alpha_2\, \mathbf{C}]\; \Gamma, \tag{5.6}$$

where $\mathbb{1}$ is the identity matrix and the symmetric matrix $\mathbf{C}$ is defined as

$$\mathbf{C} = \left\langle \psi(c(\boldsymbol{w}), y) \boldsymbol{x}\, \boldsymbol{x}^\top \right\rangle. \tag{5.7}$$

Note that μ in Eq. (5.6) counts only non-zero updates of the matrix Γ, i.e. only updates from examples assigned to $\boldsymbol{w}$. Under the assumption of constant prototype positions and stationary assignments, $\mathbf{C}$ itself remains constant in the iteration. It is important to point out that, in general, $\mathbf{C}$ cannot be interpreted as a positive (semi-) definite covariance matrix, since it takes into account label information: examples can contribute with positive or negative sign ψ.

We assume that a unique smallest eigenvalue of $\mathbf{C}$ exists with a corresponding ordering $\lambda_1 < \lambda_2 \leq \lambda_3 \ldots \leq \lambda_n$; for the influence of degeneracies we refer to Biehl et al. (submitted, 2010). Furthermore, we exploit the fact that the set of eigenvectors $\{\boldsymbol{v}_j\}_{j=1}^n$ of $\mathbf{C}$ forms an orthonormal basis of $\mathbb{R}^n$. An unnormalized update of the form (5.6) will be dominated by the largest eigenvalue and corresponding eigenvector of the matrix $[\mathbb{1} - \alpha_2 \mathbf{C}]$. For sufficiently small α_2, this eigenvalue is $(1 - \alpha_2 \lambda_1) > 0$, where λ_1 is the smallest eigenvalue of $\mathbf{C}$. However, the naive iteration of Eq. (5.6) without normalization would yield either divergent behavior for $\lambda_1 < 0$ or the trivial stationary solution $\Gamma \to 0$ with $\mu \to \infty$ for $\lambda_1 > 0$. In order to take the normalization into account explicitly, we rewrite Eq. (5.6) element-wise (omitting μ as an argument)

$$\Gamma_{ij} \leftarrow \Gamma_{ij} - \alpha_2 \, (\mathbf{C}\,\Gamma)_{ij} \tag{5.8}$$

and assume that the previous Γ was normalized. In this case, we obtain

$$\sum_{ij} \Gamma_{ij}^2 \leftarrow 1 \; - 2\alpha_2 \sum_{ij} \Gamma_{ij} (\mathbf{C}\,\Gamma)_{ij} + \; \alpha_2^2 \sum_{ij} (\mathbf{C}\,\Gamma)_{ij}^2. \tag{5.9}$$

For small learning rate α_2, the last term in Eq. (5.9) can be neglected. With the abbreviation $\kappa \equiv \sum_{ij} \Gamma_{ij} (\mathbf{C}\,\Gamma)_{ij} = \sum_{ij} \Gamma_{ji}^\top (\mathbf{C}\,\Gamma)_{ij} = \mathrm{Tr}(\,\Gamma^\top \mathbf{C}\,\Gamma\,)$, the normalized update of Γ becomes

$$\Gamma \leftarrow \frac{\Gamma - \alpha_2 \, \mathbf{C}\,\Gamma}{\sqrt{1 - 2\,\alpha_2\,\kappa}} \approx (\Gamma - \alpha_2 \, \mathbf{C}\,\Gamma)(1 + \alpha_2\,\kappa) \approx \Gamma - \alpha_2 [\mathbf{C}\,\Gamma - \kappa\,\Gamma], \tag{5.10}$$

where we neglect terms of order $\mathcal{O}(\alpha_2^2)$, again. Note that the matrix $\mathbf{C}$ appears only linearly in (5.10). Hence, applying the normalization $\sum_{ij} \Gamma_{ij}^2$ before taking the average over the subset of examples would have lead to the same result in the limit $\alpha_2 \to 0$. Eq. (5.10) shows that the stationarity condition for the matrix update is of the form

$$\mathbf{C}\,\Gamma = \kappa\,\Gamma, \tag{5.11}$$

implying that each column of the stationary matrix Γ is a multiple of some eigenvector $\boldsymbol{v}_j$ of $\mathbf{C}$. As we detail in Appendix 5.A, one can show that all columns of the stationary matrix Γ are multiples of the same eigenvector $\boldsymbol{v}_1$ corresponding to λ_1

$$\Gamma = \left[A^1\,\boldsymbol{v}_1, A^2\,\boldsymbol{v}_1, \ldots, A^n \boldsymbol{v}_1\right], \text{ with } \quad \sum_i (A^i)^2 = 1. \tag{5.12}$$

The normalization of the coefficients $A^i \in \mathbb{R}^n$ reflects that $\sum_{ij} \Gamma_{ij}^2 = 1$. Note that, apart from the specific normalization which couples the columns, the update is

equivalent to a von Mises iteration (Boehm and Prautzsch, 1993) with respect to the matrix $[\mathbb{1} - \alpha_2 \mathbf{C}]$. Hence, the update displays linear convergence and the actual convergence speed depends on the gap separating the two largest eigenvalues of $[\mathbb{1} - \alpha_2 \mathbf{C}]$ (Boehm and Prautzsch, 1993).

Finally, we can work out the resulting matrix Λ obtained from the above solution:

$$\Lambda_{ij} = \left(\Gamma\Gamma^\top\right)_{ij} = \sum_k \Gamma_{ik}\Gamma_{jk} = \sum_l (A^l)^2 v_1^i v_1^j = \left(\boldsymbol{v}_1 \boldsymbol{v}_1^\top\right)_{ij}. \tag{5.13}$$

The resulting relevance matrix is $\Lambda = \Gamma\Gamma^\top = \boldsymbol{v}_1\boldsymbol{v}_1^\top$, given by the eigenvector of $\mathbf{C}$ corresponding to the smallest eigenvalue λ_1. Note that the j-th column of Λ is given by of $v_1^j\,\boldsymbol{v}_1$. Hence, the matrix clearly satisfies the condition $\mathbf{C}\,\Lambda = \kappa\Lambda$ which we obtain by multiplying Eq. (5.11) with $\Gamma^\top$ from the right. Under the assumptions made, Λ is indeed unique, despite the freedom in selecting the coefficients A_1^k in Γ. Note also that the only non-zero eigenvalue of the resulting matrix Λ is 1.

5.3.2 Regularized matrix updates

In the following, we analyse regularized matrix updates as specified in Eq. (5.4). Before normalization, the training sample presented in time step μ induces the update

$$\Gamma(\mu+1) \sim \Gamma(\mu) - \alpha_2 \psi(c(\boldsymbol{w}), y)\, \boldsymbol{x}\,\boldsymbol{x}^\top\,\Gamma(\mu) + \alpha_2\,\eta\,\Gamma(\mu)^{-\top}. \tag{5.14}$$

In analogy to the analysis in the previous section, we investigate the dynamics of local matrix updates on average over the relevant subset of data. Equivalent to Eq. (5.6), this results in

$$\Gamma \propto \Gamma - \alpha_2\,\mathbf{C}\,\Gamma + \alpha_2\,\eta\,\Gamma^{-\top}. \tag{5.15}$$

Rewriting Eq. (5.15) element-wise and assuming that Γ was normalized after the previous time step, we obtain

$$\sum_{ij}\Gamma_{ij}^2 \leftarrow 1 - 2\,\alpha_2 \underbrace{\sum_{ij}\Gamma_{ij}(\mathbf{C}\,\Gamma)_{ij}}_{\sum_j(\Gamma^\top\mathbf{C}\,\Gamma)_{jj}} + 2\eta\,\alpha_2 \underbrace{\sum_{ij}\Gamma_{ij}(\Gamma^{-\top})_{ij}}_{\sum_j(\Gamma\Gamma^{-1})_{jj}=Tr(\mathbb{1})=n} + \mathcal{O}(\alpha_2^2)$$

which is to be compared with Eq. (5.9). Again, neglecting all terms quadratic in α_2, the normalized update of Γ reads

$$\begin{aligned}\Gamma &\leftarrow \frac{\Gamma - \alpha_2\,\mathbf{C}\,\Gamma + \eta\,\alpha_2\,\Gamma^{-\top}}{\sqrt{1 - 2\alpha_2\sum_j(\Gamma^\top\mathbf{C}\,\Gamma)_{jj} + 2\eta\alpha_2\,n}} \\ &\approx \Gamma - \alpha_2\Big[\mathbf{C}\,\Gamma - \Big(\sum_j(\Gamma^\top\mathbf{C}\,\Gamma)_{jj} - \eta n\Big)\,\Gamma - \eta\Gamma^{-\top}\Big].\end{aligned}$$

With the abbreviation $\kappa = \sum_j (\Gamma^\top \mathbf{C}\, \Gamma)_{jj} - \eta n$, we derive the stationarity condition

$$\mathbf{C}\,\Gamma - \kappa\,\Gamma = \eta \Gamma^{-\top}. \tag{5.16}$$

Obviously, for $\eta = 0$, this reduces to the same eigenvalue problem for each column of Γ as discussed previously. For $\eta \neq 0$, multiplication with $\Gamma^\top$ yields the modified stationarity condition

$$[\mathbf{C} - \kappa\,\mathbb{1}]\,\Gamma\Gamma^\top = \eta\,\mathbb{1}, \tag{5.17}$$

with the formal solution

$$\Lambda = \Gamma\Gamma^\top = \eta\,[\mathbf{C} - \kappa\,\mathbb{1}]^{-1}. \tag{5.18}$$

The symmetric matrix $[\mathbf{C} - \kappa\,\mathbb{1}]$ is invertible as long as κ does not coincide with one of the eigenvalues λ_i of $\mathbf{C}$. Note that $[\mathbf{C} - \kappa\,\mathbb{1}]$ has the same eigenvectors $\boldsymbol{v}_i$ as $\mathbf{C}$, and the eigenvalues are $(\lambda_i - \kappa)$.

Since Λ is strictly positive definite for $\eta > 0$, we have $\kappa < \lambda_1$, where λ_1 is the smallest eigenvalue of $\mathbf{C}$. In case of degenerate eigenvalues, orthonormal bases of the corresponding eigenspaces have to be considered. In any case, we can construct the inverse explicitly:

$$\Lambda = \sum_k \frac{\eta}{\lambda_k - \kappa} \boldsymbol{v}_k \boldsymbol{v}_k^\top, \text{ with } \quad \operatorname{Tr}\Lambda = \sum_i \frac{\eta}{\lambda_i - \kappa} \stackrel{!}{=} 1. \tag{5.19}$$

The normalization constraint determines κ, given the regularization parameter η and the set of $\{\lambda_k\}$. Formally, Eq. (5.19) can also be solved for η, given κ:

$$\eta = \left(\sum_i \frac{1}{\lambda_i - \kappa} \right)^{-1} \tag{5.20}$$

which leads us to

$$\Lambda = \sum_k c_k \boldsymbol{v}_k \boldsymbol{v}_k^\top, \text{ with } \quad c_k = \left(\sum_i \frac{\lambda_k - \kappa}{\lambda_i - \kappa} \right)^{-1}. \tag{5.21}$$

For every $\kappa < \lambda_1$, the corresponding solution would be obtained by using the appropriate regularization parameter η as given above. Note that we cannot set η accordingly in practical situations, since the matrix $\mathbf{C}$ is not known prior to training. However, we can obtain further insight from the relation (5.21) by investigating several cases in greater detail. In the following discussion of these special solutions, we assume that a unique smallest λ_1 exists, for convenience.

For positive definite $\mathbf{C}$, i.e. $\lambda_1 > 0$, we can consider the special solution with $\kappa = 0$. We obtain the corresponding η and Λ from the normalization constraint as

$$\eta = \frac{1}{\mathrm{Tr}\,\mathbf{C}^{-1}}, \text{ and } \quad \Lambda = \frac{\mathbf{C}^{-1}}{\mathrm{Tr}\,\mathbf{C}^{-1}}. \tag{5.22}$$

Note that this special case can be interpreted as a Mahalonobis distance only for a well separated cluster which exclusively contains data from one class. Only in such an extreme setting with $\boldsymbol{w}$ in the cluster mean, $\mathbf{C}$ is the corresponding covariance matrix.

The case $\eta = 0$ has been treated separately in Sec. 5.3.1 There, $\kappa = \lambda_1$ and Eq. (5.19) can only be considered in the simultaneous limit $\kappa \to \lambda_1^-, \eta \to 0^+$ to recover the results. For weak regularization, i.e. small but non-zero $\eta > 0$, we expect $\kappa \approx \lambda_1$ resulting in a dominating contribution of $\boldsymbol{v}_1\boldsymbol{v}_1^\top$, whereas the coefficients of $\boldsymbol{v}_j\boldsymbol{v}_j^\top$ for $j > 1$ should be relatively small. If we make the ansatz $\kappa = \lambda_1 - \epsilon$ with small $\epsilon > 0$ for $\eta \to 0$ we obtain from Eq. (5.21) the relation $\epsilon = \eta + \mathcal{O}(\eta^2)$ and coefficients

$$\begin{aligned} c_1 &= 1 - \sum_{i \geq 2} \frac{\eta}{\lambda_i - \lambda_1} + \mathcal{O}(\eta^2), \\ c_k &= \frac{\eta}{\lambda_k - \lambda_1} + \mathcal{O}(\eta^2), \text{ for } \quad k \geq 2, \end{aligned} \tag{5.23}$$

which have to be inserted in Eq. (5.21). Neglecting terms of order $\mathcal{O}(\eta^2)$ the normalized stationary matrix Λ is given by

$$\Lambda = \left(1 - \sum_{i \geq 2} \frac{\eta}{\lambda_i - \lambda_1}\right) \boldsymbol{v}_1\boldsymbol{v}_1^\top + \sum_{i \geq 2} \frac{\eta}{\lambda_i - \lambda_1} \boldsymbol{v}_i\boldsymbol{v}_i^\top. \tag{5.24}$$

As expected, the eigendirection corresponding to λ_1 dominates the distance measure. The influence of other eigenvectors $\boldsymbol{v}_k$ increases with η and is inversely proportional to $(\lambda_k - \lambda_1)$.
Finally, we consider the limit of strong regularization, $\eta \to \infty$. Eq. (5.19) implies that $\kappa \sim -\eta\, n$ which yields

$$\Lambda = \frac{1}{n} \sum_k \boldsymbol{v}_k\boldsymbol{v}_k^\top = \mathbb{1}/n.$$

Obviously, all eigenvectors contribute equally in this limit and the distance reduces to the simple Euclidean measure apart from the normalization factor $1/n$.
We would like to point out again that the stationary Γ will not be unique for a given matrix $\mathbf{C}$. Some solution of Eq. (5.18) will be approached, depending on the actual initialization. The emerging distance measure, however, and the contribution of eigendirections $\boldsymbol{v}_k$ are uniquely described by the above results.

5.3.3 Global matrix updates

In global matrix LVQ1, one unique matrix Γ defines the distance measure employed for all data. Formally, the update is identical Eq.s (5.3) or (5.4), respectively, replacing Γ_L by one and the same Γ independent of the winning prototype. Again, we can write the matrix $\mathbf{C}$ as an empirical average, but now every example contributes to the update of Γ:

$$\mathbf{C}_{\mathrm{global}} = \frac{1}{P} \sum_{\mu=1}^{P} \sum_{j=1}^{l} \phi(\boldsymbol{\xi}_\mu, \boldsymbol{w}_j)\ \psi(c(\boldsymbol{w}_j), y_\mu)\ (\boldsymbol{\xi}_\mu - \boldsymbol{w}_j)\ (\boldsymbol{\xi}_\mu - \boldsymbol{w}_j)^\top. \tag{5.25}$$

The respective winner is singled out by the indicator function ϕ with $\phi(\boldsymbol{\xi}, \boldsymbol{w}) = 1$ if $\boldsymbol{w}$ is the prototype closest to $\boldsymbol{\xi}$ and $\phi(\boldsymbol{\xi}, \boldsymbol{w}) = 0$ else. As above, $\psi(c(\boldsymbol{w}), y) = +1\,(-1)$ if the class label $c(\boldsymbol{w}) = y$ ($c(\boldsymbol{w}) \neq y$), respectively.

Note that $\mathbf{C}_{\mathrm{global}}$ depends on Γ via the assignment of data to the prototypes. Assuming stationarity of the prototypes and the *winner configuration* we can follow the lines of the analysis for local matrix updates and obtain analogous results. Here, the eigendirections of $\mathbf{C}_{\mathrm{global}}$ reflect the overall properties of the data. While the mathematical structure of the stationary state is the same as for local distance measures, its interpretation is less obvious. In particular, the definition of $\mathbf{C}_{\mathrm{global}}$ involves several centers and does not resemble a simple covariance matrix, in general.

5.4 Validity of the assumptions in practice

In practice, the matrix $\mathbf{C}$ emerges form the training and reflects properties of the data set which cannot be controlled beforehand. For instance, the smallest eigenvalues of $\mathbf{C}$ might be almost degenerate, causing potentially very slow convergence behavior. The consequences of exact degeneracies are discussed above.

The key assumption made for the above analysis is that, in the limit of large training times, $\mathbf{C}$ can be considered stationary even though Γ still changes. The matrix $\mathbf{C}$ is defined in Eq. (5.7) as an average over the subset of data assigned to a particular prototype. As outlined earlier in the section, the assumption is clearly satisfied if the data form clusters which are well separated by gaps. If the borders of receptive fields fall into these empty regions, small changes of the metrics will not alter the assignment configuration at all.

In practical situations, classes and clusters typically overlap to a certain degree. In the limit of very sparse data sets, i.e. with only a few examples in relatively high dimension, the existence of separating gaps seems plausible and our arguments should hold.

If, on the contrary, a very large number of examples is available, we can assume that the observed data samples an underlying continuous density very well, also close to the boundaries of receptive fields. Then, a very small change of Γ by training with step size $\alpha_2 \to 0$ results in small, continuous changes of $\mathbf{C}$. The above considerations can then be interpreted as a self-consistency argument for the behavior of $\mathbf{C}$ and Γ close to stationarity of both.

Problems arise when small modifications of Γ can trigger large changes of the prototype assignment. This occurs when a considerable subset of the training data is assigned to a different prototype after the training step. Such a change of the configuration can trigger seemingly discontinuous changes of the matrix $\mathbf{C}$ and its relevant eigenvalues and eigenvectors in the course of training. As a consequence, $\mathbf{C}$ can be considered approximately constant over only a limited number of training steps and it might change before the corresponding stationarity of Γ is reached. In practice, one might observe a series of transients approaching states given by Eq. (5.11) or (5.18) governed by sudden changes of $\mathbf{C}$. However, the qualitative explanation of the behavior of matrix updates remains valid, including the tendency to yield low rank relevance matrices in unregularized training.

5.5 Experiments

In the following, we illustrate our findings by means of practical simulations of localized MLVQ1. To this end, we apply the algorithm to artificial toy data and to a real world problem. The observed relevance matrices are compared to the theoretical predictions presented in the previous section. In order to quantify the agreement, we compute the Frobenius norm of $L_j = \Lambda_{\mathrm{th},j} - \Lambda_j$, where $\Lambda_{\mathrm{th},j}$ is the theoretical stationary solution and Λ_j is the observed local relevance matrix assigned to prototype $\boldsymbol{w}_j$. The Frobenius norm $\|L\|_F$ of the quadratic matrix $L \in \mathbb{R}^{n\times n}$ is given as

$$\|L\|_F = \left(\sum_{i,j=1}^{n} L_{ij}^2 \right)^{1/2} . \tag{5.26}$$

Note that for the determination of $\Lambda_{\mathrm{th},j}$, we have to compute the corresponding matrices $\mathbf{C}$ which reflect the assignment of data to the prototypes. Only after $\mathbf{C}$ has become stationary we can expect that $\|L_j\| \approx 0$. In all experiments we use the learning rate schedule specified in Eq. (3.21) for $\alpha_{1,2}$ respectively and choose the parameter τ for every application individually.

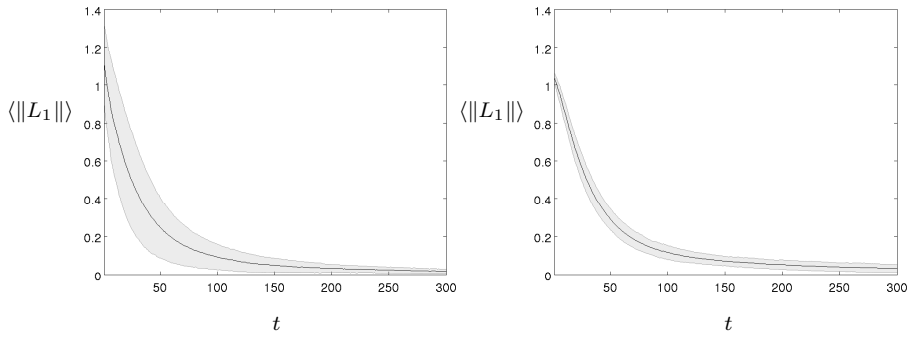
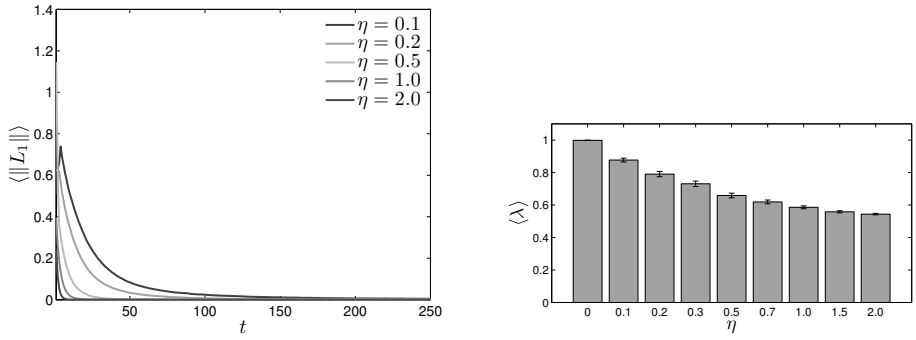


Figure 5.1: Artificial data. *Unregularized local MLVQ1. Evolution of $\|L_1\|_F$ in the course of training, averaged over ten randomized runs with two-dimensional (left panel) and ten-dimensional data (right panel), respectively. The gray shaded areas mark the observed standard deviations.*

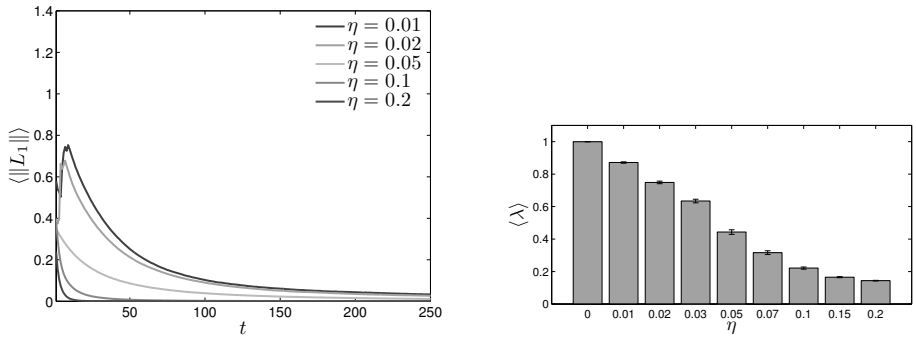
Artificial data

First, we consider a simple two-class problem in two dimensions. The data set is generated according to a mixture of two axis parallel, elongated Gaussian clusters with slight overlap. The cluster means are $\mu_1 = [-1,\, 0]$, $\mu_2 = [1,\, 0]$ while the variances are the same in both clusters: $\sigma^2_{1,2} = [0.3,\, 1.0]$. Furthermore, we embed this data set in a ten-dimensional space by adding eight feature components containing unit variance Gaussian noise. We run local MLVQ1 with the learning parameter settings $\alpha_1 = 0.01$, $\alpha_2 = 2 \cdot 10^{-4}$ and $\tau = 0.05$ for 300 epochs in each run. We train one prototype per class and initialize each prototype in the mean of random subsets of data selected from the corresponding class. Hence, the prototypes are close to the sample means, initially. Local relevance matrices are trained starting from randomized initial conditions: all elements of the matrices Γ_j are drawn from the interval $[-1, 1]$ according to a uniform density. Thereafter, a normalization step ensures $\mathrm{Tr}\,\Lambda_j = 1$. In the following, we report results for the class 1 matrix Λ_1 only as its behavior is representative for Λ_2 as well.

The training is repeated over 10 independently generated data sets containing 500 examples from each cluster. Results are presented in Fig. 5.1 in terms of the Frobenius norm $\|L_1\|_F$ on average over the randomized runs. The local relevance matrices approach the stationary state as predicted by the theory. Note that we observe that the matrices $\Lambda_{1,2}$ become singular even earlier in the training process, here after about 100 epochs. Our theoretical prediction, however, is only meaningful with respect to the stationary state of the system.



(a) Two-dimensional data set.



(b) Ten-dimensional data set.

Figure 5.2: Artificial data. *Regularized local MLVQ1. The left panels show the averaged evolution of $\|L_1\|$ for different choices of the regularization parameter η. In the right panels, the largest eigenvalue λ of the stationary Λ_1 is shown for the same values of η. All results are averaged over 10 randomized training runs and the bars in the right panels mark the corresponding standard deviations.*

Next we apply regularized matrix learning as specified in Eq. (5.4). We use the same data sets and identical learning parameters to inspect five example settings of the regularization: $\eta = 0.1, 0.2, 0.5, 1.0, 2.0$. In order to derive the corresponding Λ_{th}, we first solve Eq. (5.19) for κ given η, which can be done analytically in $n = 2$ dimensions. Afterwards, we determine the eigenvalues of $\mathbf{C}$ and compute Λ_{th} according to Eq. (5.18).

Again, we present results in terms of $\langle \|L_1\|_F \rangle$, as depicted in Fig. 5.2 (left panels). The observations confirm that the relevance matrices converge to the state specified by Eq. (5.18). Convergence appears to be faster for larger values of η. Fig. 5.2

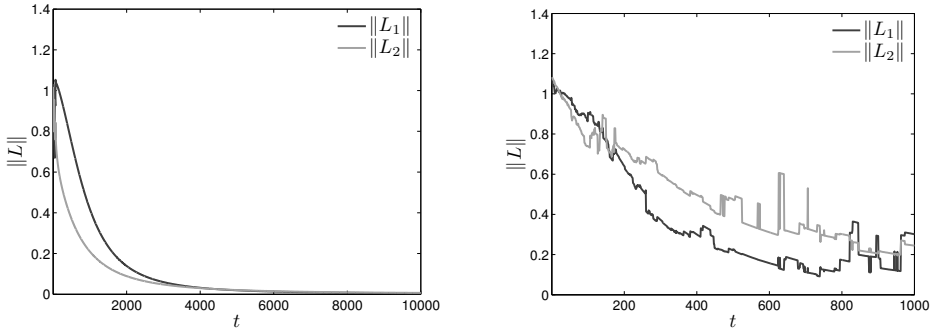


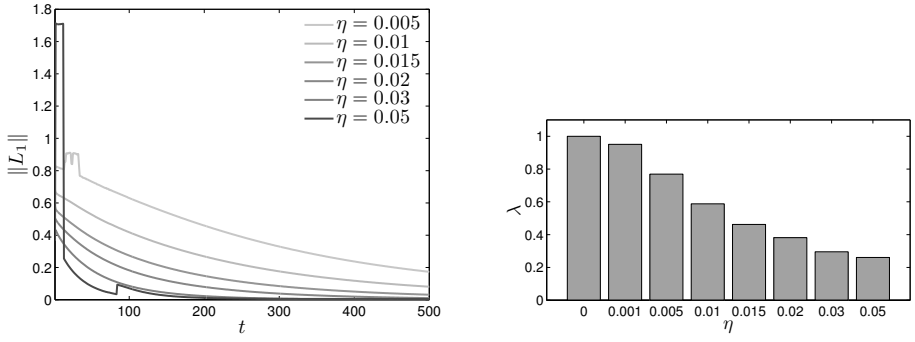
Figure 5.3: Pima data set. *Evolution of $\|L_{1,2}\|$ in unregularized local MLVQ1 with time as measured in epochs in a single example run of the training process. The left panel corresponds to a reduced data set of ten randomly selected examples per class, whereas the right panel shows results for the full data set.*

(right panels) furthermore displays the means of the largest eigenvalue of the stationary Λ_1 for several values of η. Note that for very large η all eigenvalues become equal and approach the value $1/n$ as expected.

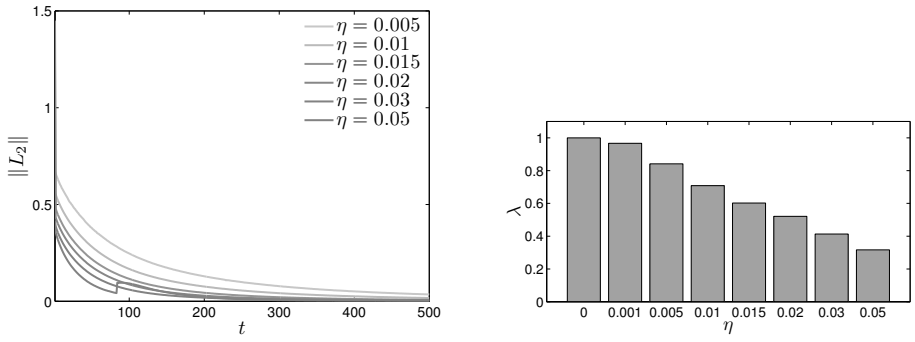
Real life data

Finally, we use a more complex real life data set from the UCI-Repository of Machine Learning to investigate the stationarity of local MLVQ1. As an illustrative example we have selected the Pima Indians diabetes data set. It constitutes a two-class problem in an eight-dimensional space. In the classification, it has to be determined whether a female of Pima Indian heritage shows signs of diabetes according to the World Health Organization criteria. The data sets consists of 500 class 1 (no diabetes) and 268 class 2 (diabetes) samples. A z-transformation is applied as a preprocessing step to normalize the data to zero mean and unit variance features. We employ local MLVQ1 with the learning rate parameters $\alpha_1 = 0.05$, $\alpha_2 = 5 \cdot 10^{-3}$, and $\tau = 0.001$, cf. Eq. (3.21). Relevance matrices are initialized randomly as described in the previous subsection. For a first set of experiments, we reduce the data set to ten randomly selected samples per class and perform 20 independent experiments. By considering only very few training examples, we can safely assume that their assignment to the prototypes becomes stationary.

For a first set of experiments, we reduce the data set to ten randomly selected samples per class and perform 20 randomized runs of unregularized training. By considering only very few training examples, we can safely assume that their as-



(a) Class 1 matrix Λ_1



(b) Class 2 matrix Λ_2

Figure 5.4: Pima data set. *The left panels show the evolution of $\|L_{1,2}\|$ in regularized local MLVQ1 with time as measured in epochs in a single example run of the training process in the reduced data set with ten examples per class for several values of the regularization parameter. The right panel displays the largest eigenvalue of $\Lambda_{1,2}$ as a function of η.*

signment to the prototypes becomes stationary. Slight modifications of the receptive fields due to the adaptation of the Γ-matrices most likely leave the assignment configuration unchanged in a very sparse data set. In this case, experiments confirm the theory very well as depicted in Fig. 5.3 (left panel).

If the full data set is employed for training, cf. Fig. 5.3 (right panel), seemingly discontinuous jumps can be observed in the experiments. Upon closer inspection, we find that the receptive fields change occasionally, leading to the phenomenon discussed in Sec. 5.4. The non-stationary assignment of the data to the prototypes results in sudden changes of the relevant eigenvectors of $\mathbf{C}_{1,2}$ and, hence, the observed relevance matrices can differ significantly from the prediction for a number

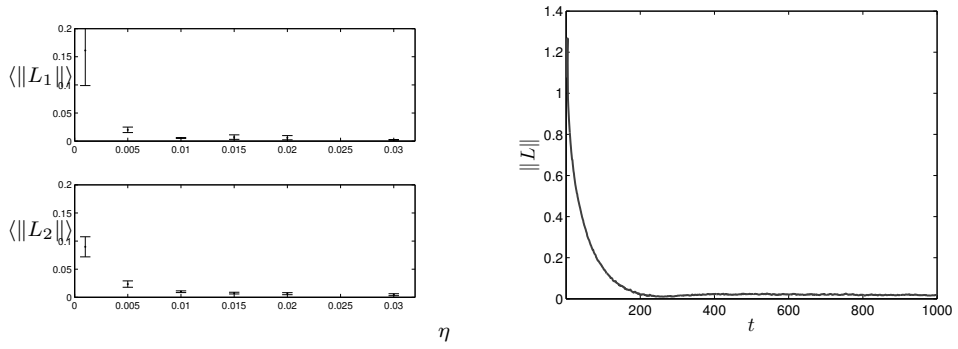


Figure 5.5: Pima data set *with ten examples per class. Left panel: The values of* $\langle\|L_{1,2}\|\rangle$ *in regularized local MLVQ1 after 10 000 epochs of training as a function of the regularization parameter* η*. Symbols display the mean over 20 randomized training runs, bars correspond to the observed standard error of mean. Right panel: Evolution of the deviation* $\|L\|$ *for a global relevance matrix with the number of epochs in an example run of unregularized global MLVQ1 training.*

of epochs.

Figure 5.4 displays the convergence behavior for the regularized version of local MLVQ1 for various regularization parameters. The jumps in $\|L_{1,2}\|$ during the initial phase of training are due to sudden changes of the prototype assignment and/or numerical problems when solving Eq. (5.19) for κ. Note, however, that the theoretical prediction concerns only the stationary state of training. For all η, we observe a clear decrease of $\|L_{1,2}\|$ with larger number of epochs. After $10\,000$ epochs, we observe for very small deviations of Λ from the predicted stationary solution, as displayed in Fig. 5.5 (left panel). Only for the smallest values of the regularization parameter η, discontinuous changes of the prototype assignment persist for very large times and result in a very slow convergence, effectively.

Finally we have tested the validity of our prediction for a single, global relevance matrix in the Pima data set. Our experiments show the same level of agreement as for localized matrix updates. Despite the more complex form of $\mathbf{C}_{\mathrm{global}}$, cf. Eq. (5.25), the behavior is analogous to the more clear-cut situation in local MLVQ1. Figure 5.5 (right panel) displays the evolution of $\|L\|$ for a global matrix in an example run.

5.6 Conclusion

We have analysed the stationary behavior of a class of matrix updates in Relevance Learning Vector Quantization. We have exemplified the treatment in terms of the extension of basic LVQ1 by local relevance matrices. Our considerations show that unregularized matrix updates tend to select a single relevant direction or a low-dimensional relevant subspace.

The matrix $\mathbf{C}$ which governs the convergence properties depends, of course, on the positioning of prototypes and implicitly on the distance measures emerging from the learning process. As a consequence, our results do not imply the possibility of obtaining the distance measure beforehand and independent of the LVQ training. In general, the relevant matrix $\mathbf{C}$ and its eigenvectors cannot be directly constructed from the data set as they emerge from the complex interplay between prototype positions and distance measure in the course of training.

For the same reason, the stationary distance measure, as specified by the matrix Λ, can depend on the initialization of the LVQ system in practical learning. The stationary winner configuration and, thus, the matrix $\mathbf{C}$ and its properties can vary between randomized learning process in the same data set.

In the extreme case of a well-separated, pure cluster of data from class $c(\boldsymbol{w})$ only, $\mathbf{C}$ is the positive definite covariance matrix with respect to $\boldsymbol{w}$ which will be approximately in the center of the cluster. In such a setting, training singles out the direction or directions of minimal variance. Clearly, the smallest distances are measured along this direction, which favors correct classification. In the more general case of separated but mixed clusters, the eigendirections selected in the training do not only correspond to its geometric properties but also reflect the cluster composition. Then, the aim of obtaining small distances for data from class $y = c(\boldsymbol{w})$ competes with the objective of achieving large distances for examples from the other classes.

One can argue that, after successful training of the prototypes, the majority of data assigned to prototype $\boldsymbol{w}$ should belong to the same class $y = c(\boldsymbol{w})$. For well-behaved data sets and if only a few examples contribute with $\psi(c(\boldsymbol{w}), y) = -1$, the relevant matrix $\mathbf{C}$ will still be positive definite, typically. Note, however, that it is not necessary to make this assumption for the arguments presented here.

In summary, the results indicate that unregularized matrix updates yield matrices Γ and Λ of small rank. In the absence of degeneracies, the distance measure takes into account a single eigendirection of $\mathbf{C}$, only. This implies that the effective number of degrees of free parameters reduces drastically from $\mathcal{O}(n^2)$ in the full matrix to $\mathcal{O}(n)$ in the stationary solution.

On the one hand, the rank reduction explains the observation that over-fitting does not seem to play a critical role in matrix relevance learning from real world data sets

(Schneider et al., 2008, 2009a). On the other hand, over-simplification effects can be due to the selection of a single relevant direction in feature space.

Within the same formal framework, we have also analysed a regularized version of matrix updates which overcomes this potential problem (Schneider et al., 2010). It is shown that a proper regularization enforces non-singularity of the stationary relevance matrix. For small regularization parameter η, the resulting matrix is still dominated by the eigenvector or eigenspace corresponding to the smallest eigenvalue of $\mathbf{C}$. By choice of η, one can control the influence of the other eigendirections continuously.

The fact that the rank of Λ remains n prevents numerical instabilities and over-simplified classification. Which choice of η gives the best learning and generalization behavior depends, of course, on details of the problem at hand and properties of the data set. Standard validation procedures could be applied to optimize the parameter in practice.

In forthcoming studies we will investigate the precise stationarity conditions for several prototype based algorithms, including cost-function oriented variants of LVQ and schemes for unsupervised learning, (e.g. Strickert et al., 2007). The mathematical structure of various matrix based prescriptions is similar, so one can expect that low rank distance measures will occur in many similarity based learning schemes.

5.A Appendix: Stationarity of unrestricted matrix updates

We consider the non-degenerate case of a unique smallest eigenvalue of $\mathbf{C}$, i.e. $\lambda_1 < \lambda_2 \leq \lambda_3 \ldots \leq \lambda_n$. In order to obtain the stationary solution of the matrix update in Eq. (5.3) we use the following ansatz for the columns of Γ:

$$\Gamma = [z_1, z_2, \ldots, z_n], \text{ with } \quad z_p = \sum_j a_j^p \boldsymbol{v}_j.$$

This is possible, since the $\{\boldsymbol{v}_j\}_{j=1}^n$ provide an orthonormal basis of $\mathbb{R}^n$. Inserting this ansatz into Eq. (5.10) allows to derive an update scheme for column $z_p(\mu) = [\Gamma(\mu)]_p$:

$$\underbrace{\sum_j a_j^p(\mu+1)\boldsymbol{v}_j}_{[\Gamma(\mu+1)]_p} = \underbrace{\sum_j a_j^p(\mu)\boldsymbol{v}_j}_{[\Gamma(\mu)]_p} - \alpha_2 \Big[\underbrace{\sum_j a_j^p(\mu)\lambda_j \boldsymbol{v}_j}_{[\mathbf{C}\,\Gamma(\mu)]_p} - \kappa(\mu) \underbrace{\sum_j a_j^p(\mu)\boldsymbol{v}_j}_{[\Gamma(\mu)]_p} \Big].$$

Due to the orthogonality of eigenvectors, this corresponds to the evolution of coefficients

$$a_j^p(\mu+1) = a_j^p(\mu)\left[1 - \alpha_2\ (\lambda_j - \kappa(\mu))\right]. \tag{5.27}$$

Let $\tilde{\kappa}$ and A_j^p be the stationary values of κ and a_j^p, respectively. Since a unique value of $\tilde{\kappa}$ has to satisfy $0 = A_j^p(\lambda_j - \tilde{\kappa})$ for all j, the only non-trivial solution corresponds to $\tilde{\kappa} = \lambda_p$ for one particular m with $A_q^p \neq 0$, but $A_j^p = 0$ for all $j \neq q$.
Consequently, Eq. (5.27) reads close to stationarity

$$a_j^p(\mu+1) \approx a_j^p(\mu) \cdot \left[1 - \alpha_2(\lambda_j - \lambda_q)\right]. \tag{5.28}$$

Now, let us assume that $q > 1$. In this case, the factor $[1 - \alpha_2\,(\lambda_j - \lambda_q)] > 1$ for all $j < q$, because $\lambda_j < \lambda_q$. Small deviations of the corresponding a_j^p from $A_j^p = 0$ would grow in the iteration, indicating an instability. Hence, the only stable, consistent solution is $q = 1,\ \tilde{\kappa} = \lambda_1,\ A_1^p \neq 0,\ A_j^p = 0$, for all $j > 1$. This leads immediately to Eq. (5.12), where we omit the subscript 1 of the A_1^p for brevity.

Material based on:

Petra Schneider, Michael Biehl and Barbara Hammer - *"Hyperparameter Learning in Probabilistic Prototype-based models,"* Neurocomputing, vol. 73 , no. 7-9, 2010.

Petra Schneider, Tina Geweniger, Frank-Michael Schleif, Michael Biehl and Thomas Villmann - *"Generalizing Robust Soft LVQ to handle data with uncertain class labels,"* submitted, 2010.

Chapter 6

Hyperparameter learning in probabilistic prototype-based models

Abstract

We present approaches to extend Robust Soft Learning Vector Quantization (RSLVQ). This algorithm for nearest prototype classification is derived from an explicit cost function and follows the dynamics of a stochastic gradient ascent. The RSLVQ cost function is defined in terms of a likelihood ratio and involves a hyperparameter which is kept constant during training. We propose to adapt the hyperparameter in the training phase based on the gradient information. Besides, we propose to base the classifier's decision on the value of the likelihood ratio instead of using the distance based classification approach. Experiments on artificial and real life data show that the hyperparameter crucially influences the performance of RSLVQ. However, it is not possible to estimate the best value from the data prior to learning. We show that the proposed variant of RSLVQ is very robust with respect to the initial value of the hyperparameter. The classification approach based on the likelihood ratio turns out to be superior to distance based classification, if local hyperparameters are adapted for each prototype. Moreover, we generalize RSLVQ with respect to the treatment of vectorial class labels in the training data. This allows to integrate uncertain class information of training data into the learning process of an RSLVQ classifier.

6.1 Introduction

The learning rules of several LVQ procedures involve a hyperparameter, such as the window size in LVQ2.1 (Kohonen, 1997) or the softness parameter σ^2 in Soft LVQ (Seo and Obermayer, 2003) and Robust Soft LVQ (Seo and Obermayer, 2003). The hyperparameter can have high impact on the performance of the resulting classifier. Usually, it is kept constant in the learning process, and it is chosen from a set of

candidates by means of a validation procedure. In Seo and Obermayer (2006), an annealing schedule is proposed to reduce the respective hyperparameter of an LVQ algorithm in the course of training. However, this schedule is purely heuristically motivated and does not follow any learning objective.

This work focuses on RSLVQ (see Sec. 2.3.3) which is based on a well defined stochastic model of LVQ classification schemes. Training is based on the objective of likelihood optimization. The learning rules are derived from an explicit cost function which is optimized with respect to the model parameters. In this study, we introduce a well-founded strategy to deal with the hyperparameter, and we propose a new decision rule to classify the data. Since the cost function also depends on σ^2, we propose to introduce the hyperparameter as a further degree of freedom and to maximize the objective function with respect to σ^2 as well. Further, we compare the performances of simple nearest prototype classification and schemes which are explicitly based on the likelihood. The latter corresponds naturally to the objective of training.

Finally, we generalize the RSLVQ cost function with respect to vectorial class labels for training data. In particular, we suppose that each element of the label vectors is in the range $[0, 1]$ describing the fuzzy assignment of the data to the respective class. This approach allows to consider insecure label information for the construction of a nearest prototype classifier.

In our experiments, we illustrate the influence of the hyperparameter on the classification accuracy of RSLVQ and study the effect of the proposed optimization method using artificial data and real-life data sets. Further, we show the superiority of likelihood based classification in particular for local adaptation schemes. Based on artificial data, we demonstrate that the extended cost function yields a generalization of the original RSLVQ algorithm.

6.2 Modifications of Robust Soft LVQ

The derivation of the update rules of the algorithms in Sec.s 6.2.1 and 6.2.3 are based on the assumption of a Gaussian mixture model. Hence, the densities $p(\boldsymbol{\xi}|j)$ in Eq. (2.8) have the normalized exponential form (see Eq. (2.9)) with

$$K(j) = \frac{1}{(2\pi\sigma_j^2)^{n/2}}, \qquad f(\boldsymbol{\xi}, \boldsymbol{w}_j, \sigma_j^2) = -\frac{(\boldsymbol{\xi} - \boldsymbol{w}_j)^T(\boldsymbol{\xi} - \boldsymbol{w}_j)}{2\,\sigma_j^2}. \tag{6.1}$$

The derivatives constitute

$$\frac{\partial f(\boldsymbol{\xi}, \boldsymbol{w}_j, \sigma_j^2)}{\partial \boldsymbol{w}_j} = \frac{1}{\sigma^2}(\boldsymbol{\xi} - \boldsymbol{w}_j), \tag{6.2}$$

$$\frac{\partial f(\boldsymbol{\xi}, \boldsymbol{w}_j, \sigma_j^2)}{\partial \sigma_j^2} = \frac{(\boldsymbol{\xi} - \boldsymbol{w}_j)^T (\boldsymbol{\xi} - \boldsymbol{w}_j)}{2\,\sigma_j^4}, \tag{6.3}$$

$$\frac{\partial K(j)}{\partial \sigma_j^2} = -\frac{n}{2}\,\frac{1}{(2\pi\sigma_j^2)^{n/2}\,\sigma_j^2}. \tag{6.4}$$

6.2.1 Hyperparameter adaptation in RSLVQ

In Seo and Obermayer (2006), a heuristic approach is introduced to anneal the value of the hyperparameter in the course of training. The authors propose a schedule which reduces σ^2 continuously in each learning step. This may lead to non-monotonic learning curves, as the performance deteriorates when σ^2 becomes lower than the potential optimum. Hence, the method has to be used in combination with an early stopping procedure.

In this work, we propose a more systematic approach to treat the hyperparameter according to the optimization of the likelihood ratio in Eq. (2.12). We adapt the hyperparameter according to the gradient of E_{RSLVQ} with respect to σ^2. In case of a Gaussian mixture model, the derivatives in Eq. (3.27) and Eq. (6.3) lead us to the update rule

$$\Delta\sigma^2 = \alpha_2 \sum_j \Bigg(\Big(\delta_{y,c(\boldsymbol{w}_j)}\,(P_y(j|\boldsymbol{\xi}) - P(j|\boldsymbol{\xi})) - (1 - \delta_{y,c(\boldsymbol{w}_j)})\,P(j|\boldsymbol{\xi}) \Big) \cdot \frac{d(\boldsymbol{\xi}, \boldsymbol{w}_j)}{\sigma^4} \Bigg). \tag{6.5}$$

The Kronecker symbol $\delta_{y,c(\boldsymbol{w}_j)}$ tests whether the labels $c(\boldsymbol{w}_j)$ and y coincide, and $\alpha_2 > 0$ is the learning rate. The method becomes even more flexible by training an individual hyperparameter σ_j^2 for every prototype $\boldsymbol{w}_j$. Due to the derivative in Sec. 3.A.2 and Eq.s (6.3), (6.4) we obtain the learning rule

$$\Delta\sigma_j^2 = \frac{\alpha_2}{\sigma_j^2} \cdot \begin{cases} (P_y(j|\boldsymbol{\xi}) - P(j|\boldsymbol{\xi}))\,(-n + d(\boldsymbol{\xi}, \boldsymbol{w}_j)/\sigma_j^2), & c(\boldsymbol{w}_j) = y, \\ -P(j|\boldsymbol{\xi})\,(-n + d(\boldsymbol{\xi}, \boldsymbol{w}_j)/\sigma_j^2), & c(\boldsymbol{w}_j) \neq y. \end{cases} \tag{6.6}$$

Using this approach, the update rules for the prototypes (see Seo and Obermayer, 2003) also include the local hyperparameters σ_j^2.

Note that σ^2 or σ_j^2, respectively, play the role of the variance of local Gaussians when modelling the distribution of the probability density of data in this likelihood framework. As such, the question occurs whether these parameters converge to the variance underlying the data distribution which would allow a prior estimation of these parameters from the data. We will show in experiments that this is not the case in general. The optimum choice of σ^2 is subtle, and an automatic adaptation

scheme as proposed in this work constitutes the method of choice. These issues arise from two reasons: on the one hand, the variance is optimized within a discriminative framework such that values which do not coincide with the underlying data distribution might be favorable. Further, the variance controls the influence of training points on the adaptation scheme and it determines the size of the region of interest during training. Unlike heuristics, such as window schemes in LVQ2.1 (Kohonen, 1997), automatic adaptation provides a principled way to optimize these parameters.

6.2.2 Decision rule based on likelihood ratio

Beyond the new strategy for dealing with the parameter σ^2, we propose to base the classification on the likelihood ratio defined in Eq. (2.11). The closest prototype scheme as described in Sec. 2.2 is replaced by a highest likelihood ratio classification, i.e. a feature vector $\boldsymbol{\xi}$ is assigned to class i with $L(\boldsymbol{\xi}, i) > L(\boldsymbol{\xi}, j), \forall j \neq i$. Note that the two different approaches lead to the same decision in case of LVQ-systems with one prototype per class and a global hyperparameter σ^2. However, different classification results can be obtained in case of a larger number of prototypes per class and/or training of individual hyperparameters for each prototype as proposed in Sec. 6.2.1.

This approach has no influence on learning rules of RSLVQ. It affects, however, the classification performance of a given system after training.

6.2.3 Generalized cost function

In the following, we provide a generalization of the RSLVQ cost function with respect to vectorial input data. The objective of this approach is take into account insecure label information of training data lying close to the decision boundary. We derive the learning rules for the prototypes and the hyperparameter σ^2 as a stochastic gradient of the extended cost function. We term the novel algorithm Fuzzy Robust Soft LVQ. In the limit of crisp class memberships, the cost function and the updates are equivalent to the original RSLVQ algorithm. Note that the resulting classifier performs a crisp classification although fuzzy labeled data is used for the training process.

Every training pattern $\boldsymbol{\xi}$ carries a probabilistic label vector $\boldsymbol{y} \in [0, 1]^C$, with $\sum_i y^i = 1$. Component y^i constitutes the assignment probability of sample $\boldsymbol{\xi}$ to class i. Since the classifier performs a crisp classification, the prototypes still represent exactly one class. Under this assumption, the definition of $\mathbf{W}$ (see Eq. (2.1))) remains unchanged compared to the crisp version of the algorithm. The same holds for the

definition of the probability density $p(\boldsymbol{\xi}|\mathbf{W})$ in Eq. (2.8). Since $p(\boldsymbol{\xi}|\mathbf{W})$ only takes the prototypes' class memberships into account, the density remains unchanged due to the fuzzy labeling of the data. However, the class specific density in Eq. (2.10) needs to be generalized in order to handle data with vectorial class labels. This generalization constitutes the weighted sum of the class-specific densities; the weight values are given by the assignment probabilities to the different classes

$$p(\boldsymbol{\xi}, \boldsymbol{y}|\mathbf{W}) = \sum_{i=1}^{C} y^i \sum_{j:c(\boldsymbol{w}_j)=i}^{m} p(\boldsymbol{\xi}|j)P(j). \tag{6.7}$$

The cost function of Fuzzy RSLVQ arises out of E_{RSLVQ} by defining the likelihood ratio in Eq. (2.11) in terms of $p(\boldsymbol{\xi}, \boldsymbol{y}|\mathbf{W})$.

In accordance with original RSLVQ (Seo and Obermayer, 2003), we implement the optimization of E_{FRSLVQ} in terms of a stochastic gradient ascent. The derivative of the novel cost function with respect to the model parameters is provided in Sec. 6.A. Combining Eq.s (6.2) - (6.4) and Eq. (6.11) yields the update rules for the prototypes and the hyperparameters

$$\Delta \boldsymbol{w}_j = \frac{\alpha_1}{\sigma_j^2} \left(P_{\boldsymbol{y}}(j|\boldsymbol{\xi}) - P(j|\boldsymbol{\xi})\right) (\boldsymbol{\xi} - \boldsymbol{w}_j), \tag{6.8}$$

$$\Delta \sigma_j^2 = \alpha_2 \left(P_{\boldsymbol{y}}(j|\boldsymbol{\xi}) - P(j|\boldsymbol{\xi})\right) \left(\frac{d(\boldsymbol{\xi}, \boldsymbol{w}_j)}{\sigma_j^4} - \frac{n}{\sigma_j^2}\right), \tag{6.9}$$

where $\alpha_{1,2}$ are the learning rates. For the more general case of a global parameter $\sigma_j^2 = \sigma^2, \forall j$, the hyperparameter can be updated by the summation of the probability assignments

$$\Delta \sigma^2 = \alpha_2 \sum_{j=1}^{m} \left(P_{\boldsymbol{y}}(j|\boldsymbol{\xi}) - P(j|\boldsymbol{\xi})\right) \cdot \frac{d(\boldsymbol{\xi}, \boldsymbol{w}_j)}{\sigma^4}. \tag{6.10}$$

In the experimental section, we focus on the adaptation of a global hyperparameter σ^2 only. The Fuzzy RSLVQ algorithm is defined in terms of Eq.s (6.8) - (6.10).

Note that $(P_{\boldsymbol{y}}(j|\boldsymbol{\xi}) - P(j|\boldsymbol{\xi}))$ turns into the weight factors in the update rules of algorithms previously derived from RSLVQ, if $\boldsymbol{y}$ specifies a crisp class labeling of sample $\boldsymbol{\xi}$ (e.g. Eq.s (3.18), (3.19), (6.6)). The factor is positive for model parameters related to class j with $j = \arg\max_i (y^i)$ and negative otherwise. This is also illustrated in Fig. 6.1 by means of a one-dimensional example setting consisting of two prototypes. The plots depict the update strength $(P_{\boldsymbol{y}}(j|\boldsymbol{\xi}) - P(j|\boldsymbol{\xi}))$ caused by data samples of different fuzziness. Furthermore, the influence of the hyperparameter σ^2 is illustrated. The figures depict that $\int (P_{\boldsymbol{y}}(j|\boldsymbol{\xi}) - P(j|\boldsymbol{\xi}))\, d\boldsymbol{\xi}$ is small for highly

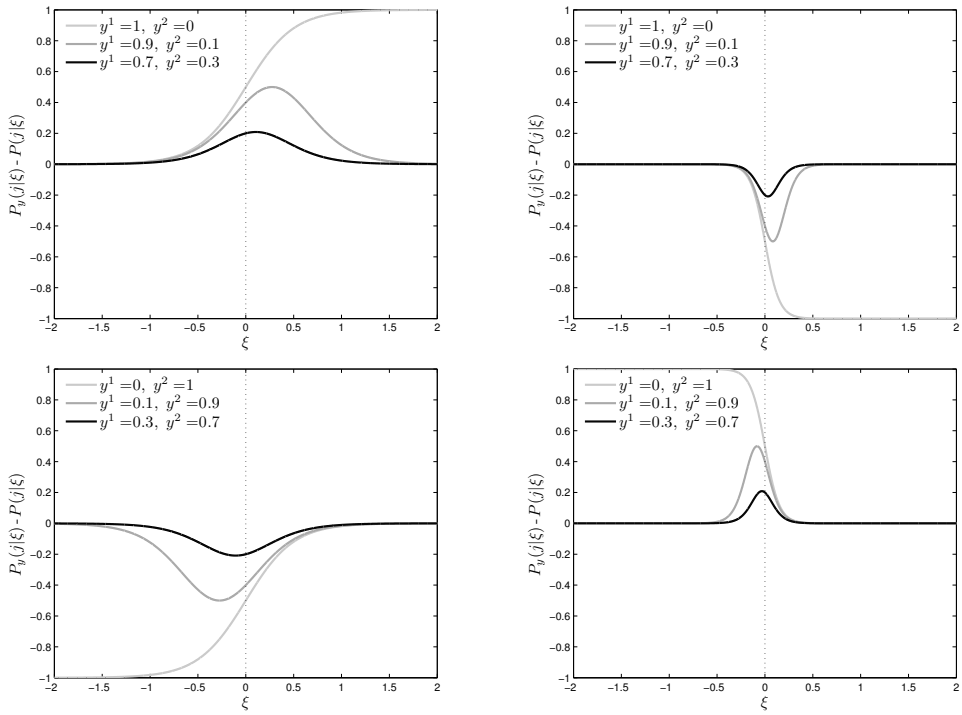


Figure 6.1: *Illustration of the weight factors* $(P_{\boldsymbol{y}}(j|\boldsymbol{\xi}) - P(j|\boldsymbol{\xi}))$ *affecting the prototypes* $\boldsymbol{w}_{1,2}$ *in a one-dimensional setting with two prototypes* $\boldsymbol{w}_1 = -1$ *and* $\boldsymbol{w}_2 = 1$*. In each figure, the curves correspond to training samples* $\boldsymbol{\xi}$ *with different fuzzy label* $\boldsymbol{y}$*.* **Left column***: Forces affecting* $\boldsymbol{w}_1$ *with* $\sigma^2 = 0.5$*. Top:* $y^1 > 0.5$ *(attractive forces). Bottom:* $y^1 < 0.5$ *(repulsive forces).* **Right column***: Forces affecting* $\boldsymbol{w}_2$ *with* $\sigma^2 = 0.15$*. Top:* $y^1 > 0.5$ *(repulsive forces). Bottom:* $y^1 < 0.5$ *(attractive forces).*

fuzzy data, but increases with decreasing fuzziness. In the limit of crisp class labeling, the integral tends to infinity. The hyperparameter determines the width of the active region. For small σ^2, the active region is very sharp and focused on the region around the decision boundary. Training samples from a larger area in feature space contribute to the training, if σ^2 is set to larger values.

6.3 Experiments

In a first set of experiments, the proposed extensions of RSLVQ are applied to artificial toy data. The data sets consist of spherical Gaussian clusters in a two-dimensional

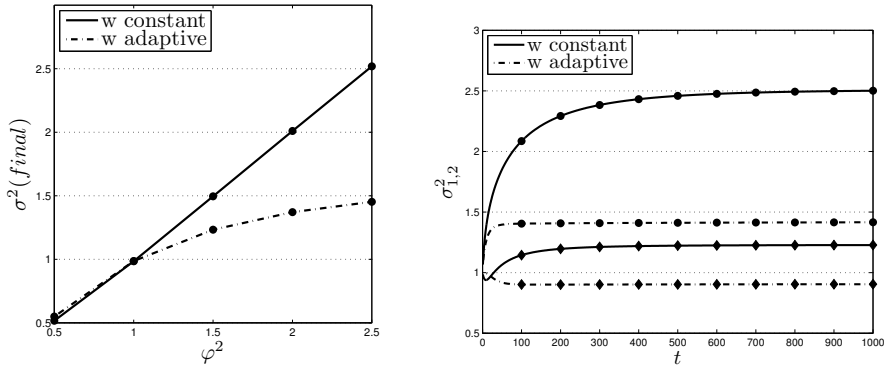


Figure 6.2: *Artificial data.* **Left:** *Variance φ^2 of the Gaussians vs. mean final value of the global hyperparameters σ^2 obtained on data sets with two clusters of equal variance and constant and adaptive prototypes.* **Right:** *Evolution of the local hyperparameters $\sigma^2_{1,2}$ as a function of training time observed on a data set of two Gaussians with unequal variance and constant and adaptive prototypes. The variances are $\varphi^2_1 = 2.5$ and $\varphi^2_2 = 1.25$. The symbols correspond to σ^2_1 (•), σ^2_2 (♦).*

space and correspond to binary classification problems. We investigate the relation between the hyperparameter σ^2 and the variance of the Gaussians for crisp and fuzzy labeled data. Furthermore, we highlight differences between the alternative decision rules based on the Euclidean distance and the likelihood ratio.

In order to evaluate the performance of the new techniques in real life situations, the methods are applied to two benchmark data sets from the UCI repository of machine learning (Newman et al., 1998).

In all experiments, the learning rates are continuously reduced with training time according to the schedule in Eq. (3.21). To initialize the prototypes, we choose the mean values of random subsets of data points selected from each class.

6.3.1 Artificial data

The adaptation of a global hyperparameter is analysed by means of data sets consisting of two Gaussian clusters of equal variance. The clusters are centered at $\mu_1 = [-2, 0]$, $\mu_2 = [2, 0]$ and consist of 1 000 data points each. The data sets differ with respect to the cluster variances φ^2 which vary between 0.5 and 2.5. At first, we fix the prototypes to the mean values of the distributions in order to analyse the adaptation of σ^2 independent of other model parameters. In the next experiments, the hyperparameter and the prototypes are optimized simultaneously. The learning

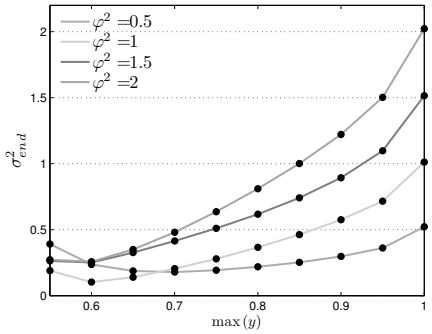


Figure 6.3: *Artificial data. Mean final value of the hyperparameter as a function of the fuzziness y_1^1 of the training data obtained on data sets with different variance φ^2.*

parameters are set to $\alpha_1 = 0.01$, $\alpha_2 = 0.001$, $c = 0.001$ and the system is trained for 1 000 epochs. The softness is initialized by $\sigma^2(0) = 1$. The results presented in the following are averaged over experiments on ten statistically independent data sets.

Fig. 6.2 (left) visualizes the mean final values of the hyperparameter obtained on the different data sets as a function of φ^2. If the prototypes are constant and are placed in the cluster centers, σ^2 converges towards the variance of the Gaussians. However, σ^2 approaches smaller values in the experiments with adaptive prototypes. Concurrently, we observe that the prototypes saturate closer to the decision boundary as φ^2 increases.

These results are also confirmed by further experiments with two spherical clusters of different variance $\varphi^2_{1,2}$ and local adaptive hyperparameter; see Fig. 6.2, right, for an example.

Hence, maximizing the likelihood ratio in Eq. (2.11) corresponds to a density estimation only if the prototypes correspond to the cluster means. However, the optimal hyperparameter cannot be estimated from the data directly, if the classifier is also optimized with respect to the prototype positions. This holds because of three reasons: for multi-modal data sets with several prototypes per class, the assignment of data to prototypes is not clear a priori, such that no statistical estimations can be made. Even for data sets which are represented using only one prototype per class, an estimation of σ^2 from the data is not obvious since, besides the bandwidth, σ^2 determines the influence of training points on the adaptation and hence, the overall dynamics. Further, prototypes do not necessarily coincide with the class centers, rather, prototype locations and bandwidth are adapted by the learning rule to give an optimum decision boundary.

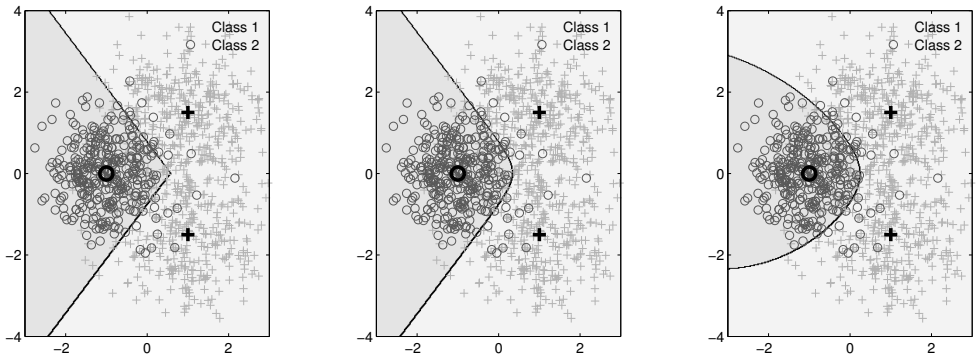


Figure 6.4: *Receptive fields induced by distance based classification and likelihood ratio based classification for a two-class problem consisting of three clusters. The cluster means serve as prototypes.* **Left:** *Distance based classification using the squared Euclidean distance.* **Middle:** *Likelihood ratio based classification after training of a global hyperparameter* σ^2. **Right:** *Likelihood ratio based classification after training of local hyperparameters* $\sigma^2_{1,2,3}$.

Furthermore, we perform equivalent experiments in a generalized framework to demonstrate that the previous observations are results of Fuzzy RSLVQ under special conditions. To generalize the previous settings, all data generated by one distribution need to carry identical label vectors $\boldsymbol{y}_1$ or $\boldsymbol{y}_2$. Moreover, the conditions $y_1^1 = y_2^2$ and $y_1^2 = y_2^1$ need to be fulfilled. In the following, we always refer to y_1^1 to specify the fuzziness of the data.

Following the same procedure, we first set the prototypes fixed in the cluster centers to analyse the adaptation of σ^2 exclusively. Afterwards, we train the prototypes with constant $\sigma^2 = 1$. We analyse, how the fuzziness of the input data influences the resulting model. We choose the learning parameter settings $\alpha_1 = 1 \cdot 10^{-4}$, $\alpha_2 = 1 \cdot 10^{-5}$ and $c = 0.001$. The hyperparameter initialized by $\sigma^2(0) = 1$.

Our observations are depicted in Fig. 6.3. According to the previous experiments, σ^2 approaches φ^2 in case of the crisp labeling $y_1^1 = 1$. Remarkably, with increasing fuzziness, σ^2 initially approaches smaller values. Note however that the curves pass a minimum. In the experiments with highly fuzzy data (y_1^1 close to 0.5), the final value $\sigma^2(t)$ increases again. The value y_1^1 which results in the minimal value σ^2 depends on φ^2. With $y_1^1 < 0.5$, σ^2 converges to very large values > 10.

If we adapt the prototypes with constant hyperparameter $\sigma^2 = 1$, $\boldsymbol{w}_{1,2}$ do not saturate in the cluster centers. The fuzziness determines the prototypes' distance to the decision boundary. In case of $y_1^1 = 1$, incorrectly classified data contribute much stronger to the value of the cost function compared to correctly classified samples.

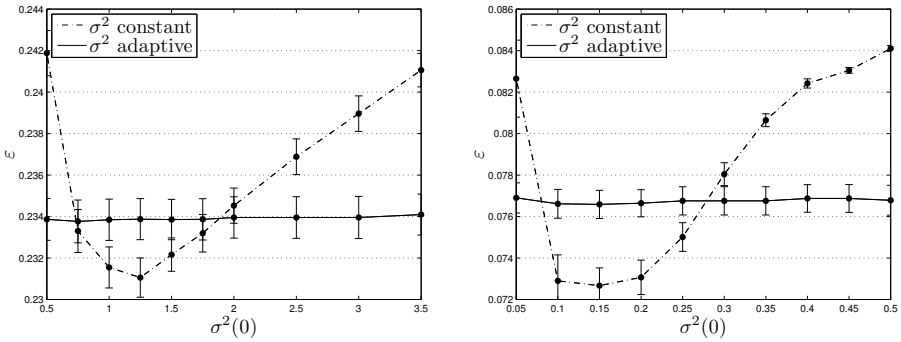


Figure 6.5: *Mean test performance at the end of RSLVQ-training with constant and adaptive hyperparameter as a function of the initial value $\sigma^2(0)$. The error bars indicate the standard error.* **Left:** Letter *data set* **Right:** Pendigits *data set.*

For this reason, the attractive forces caused by misclassified patterns on the opposite side of the decision boundary dominate the update. For this reason, the prototypes move in the direction of the decision boundary. Increasing fuzziness weakens this effect and the distance between the prototypes increases.

Finally, we compare the decision boundaries induced by nearest prototype classification and the decision rule based on the likelihood ratio. For this purpose, a data set consisting of three clusters is used; it is visualized in Fig. 6.4. The mean value of the class 1 data is $\mu_1 = [-1, 0]$. The class two data is split into two clusters centered at $\mu_2 = [1, 1.5]$ and $\mu_3 = [1, -1.5]$. The variances constitute $\varphi_1^2 = 0.5$, $\varphi_2^2 = 0.8$ and $\varphi_3^2 = 1.0$. Each cluster consists of 1 000 samples. According to the priorly known distribution, the data is approximated by three prototypes. We set the prototypes fixed to the mean values $\mu_{1,2,3}$ and adapt global and local hyperparameters. We use the learning parameter settings $\alpha_2 = 1 \cdot 10^{-5}$, $c = 1 \cdot 10^{-4}$, $\sigma^2(0) = \sigma_j^2(0) = 0.1, \forall j$ and train for 1 000 epochs. On average, the global hyperparameter saturates at $\sigma^2 \approx 0.7$. Similar to the previous experiments, the values $\sigma_{1,2,3}^2$ approach $\varphi_{1,2,3}^2$. Fig. 6.4 visualizes the receptive fields for the alternative decision rules resulting from these prototype and hyperparameter settings. Distance based classification with the squared Euclidean distance leads to piecewise linear decision boundaries. Remarkably, the receptive fields are no longer separated by straight lines, if a sample is assigned to the class of highest likelihood ratio. The effect is even more pronounced, if local softness parameters are assigned to the prototypes as displayed in the right-most panel of Fig. 6.4.

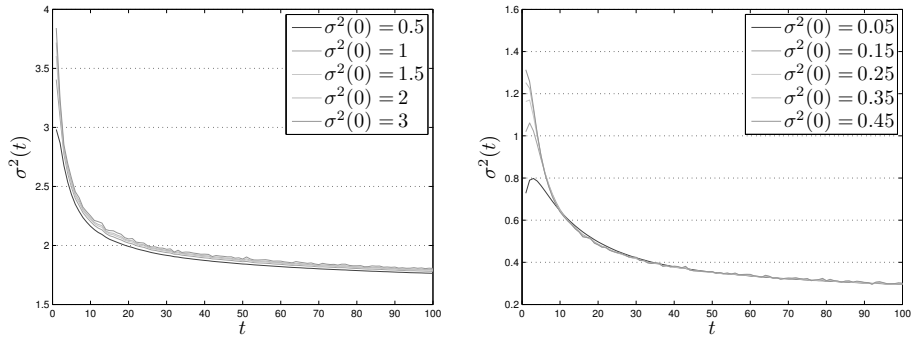


Figure 6.6: *Evolution of the global hyperparameter σ^2 as a function of training time for different initial settings $\sigma^2(0)$.* **Left:** Letter *data set.* **Right:** Pendigits *data set.*

6.3.2 Real life data

In the second set of experiments, we apply the algorithms to the Letter Recognition- and Pendigits data set provided by the UCI-Repository of Machine Learning (Newman et al., 1998).

Letter Recognition

The data set consists of 20 000 feature vectors which encode 16 numerical attributes of black-and-white rectangular pixel displays of the capital letters of the English alphabet; hence, a 26-class problem is dealt with. We split the data randomly into a training and a test set of equal size. The following results are averaged over ten different compositions of training and test data.

At first, we focus on RSLVQ-training with global σ^2 and adapt one prototype per class. Note that the two alternative approaches for classification always lead to the same decision in this case. We train prototypes and hyperparameter with different initial settings $\sigma^2(0)$ and compare the results to equivalent RSLVQ experiments without σ^2-adaptation. We choose various values $\sigma^2(0)$ from the interval [0.5, 3.5]. The remaining learning parameters are set to $\alpha_1 = 0.01$, $\alpha_2 = 0.001 \cdot \sigma^2(0)$ and $c = 0.1$. Training is continued for 100 epochs.

The experiments with constant σ^2 show that the performance of RSLVQ is highly sensitive with respect to the value of the hyperparameter (see Fig. 6.5, left). The lowest mean rate of misclassification on the test sets is achieved with $\sigma^2_{opt} = 1.25$; the performance constitutes $\varepsilon_{test} \approx 23.1\%$. However, the curve in Fig. 6.5 shows a very sharp minimum, indicating a strong dependence of the classification performance on the value of the hyperparameter. For small $\sigma^2 < 1$, we observe instabilities and highly fluctuating learning curves.

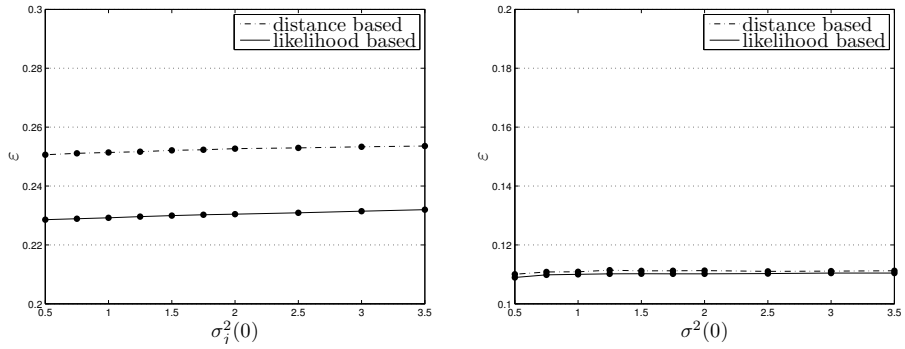


Figure 6.7: Letter *data set. Mean final test errors after training of global and local hyperparameters as a function of the initial values $\sigma^2(0)$ and $\sigma_j^2(0)$. Standard error bars would be smaller than the symbol size. The plots compare the performance yielded by the alternative decision rules for classification.* **Left:** *Experiments with local adaptive hyperparameters and one prototype per class.* **Right:** *Experiments with global adaptive hyperparameter and five prototypes per class.*

Remarkably, by including the proposed optimization scheme for global hyperparameter into the training, the sensitivity of the algorithm with respect to σ^2 can be eliminated. In all experiments with adaptive hyperparameter, the mean test error saturates at $\varepsilon_{test} \approx 23.4\%$, independent of the initial setting $\sigma^2(0)$ (see Fig. 6.5, left). Furthermore, the initialization $\sigma^2(0)$ does not influence the final value of the hyperparameter. As depicted in Fig. 6.6, left, the parameter converges towards $\sigma^2_{final} \approx 1.8$ in all experiments. Hence, the proposed variant of RSLVQ is much more robust related to the initial choice of the hyperparameter. Especially for large values $\sigma^2(0)$, the proposed optimization method achieves a clear improvement in classification performance and speed of convergence, compared to RSLVQ training with constant σ^2. However, despite the extended flexibility, learning with constant $\sigma^2 = \sigma^2_{opt}$ still achieves a slightly better performance than our method. This observation can be explained by the fact that the relation between the RSLVQ cost function and the classification performance is not obvious. The optimum of the likelihood ratio does not necessarily coincide with minimal rate of misclassification. Nevertheless, the learning strategy for σ^2 may simplify the identification of σ^2_{opt} to achieve the optimal classification performance.

The adaptation of local hyperparameters is continued for 300 epochs. The error curves converge on constant level after ca. 150 epochs. Interestingly, the classification performance differs significantly for the alternative decision rules. As depicted in Fig. 6.7, left, highest likelihood ratio classification achieves $\approx 2\%$ lower mean

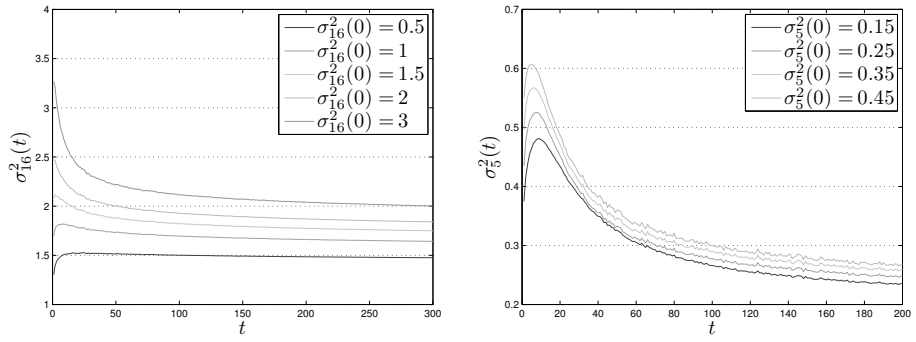


Figure 6.8: *Evolution of local hyperparameters σ_j^2 as function of training time for different initial settings $\sigma_j^2(0)$.* **Left:** Letter *data set, $j = 16$.* **Right:** Pendigits *data set, $j = 5$.*

error rate on the test samples. However, a slight dependence of the error rate on the initialization $\sigma_j^2(0)$ can be observed. In contrast to our experiments with global σ^2, the final local hyperparameters spread more for different settings $\sigma_j^2(0)$. This is visible in Fig. 6.8, left, which compares the evolution of $\sigma_{16}^2(t)$ for different initializations; the curves are representative for all σ_j^2. We expect the differences to vanish for smaller learning rates and training over a larger number of epochs. Note that the performance of distance based classification even degrades due to the adaptation of local hyperparameters, if compared to the experiments with global σ^2.

Furthermore, we analyse how the number of prototypes affects the performance of the alternative classification strategies. We train five prototypes per class and adapt a global hyperparameter. Due to the larger number of prototypes, σ^2 approaches to smaller values compared to the previous experiments; on average, it saturates at $\sigma_{final}^2 \approx 0.8$, independent of $\sigma^2(0)$. Although the classification improves significantly due to the larger number of prototypes, distance based classification and likelihood ratio based classification achieve nearly the same error rate of $\varepsilon_{test} \approx 11\%$ in all experiments (see Fig. 6.7, right).

Apparently, only training of local hyperparameters causes a significant difference between the two alternative decision rules. Compared to RSLVQ with a global hyperparameter, the adaptation of local σ_j^2 improves the performance of likelihood ratio based classification, but decreases the performance of distance based classification. In contrast, the performance of both approaches is nearly equal, if a larger number of prototypes is used in combination with a global parameter σ^2. The first observation concerning the effect of local hyperparameter adaptation is also confirmed in further experiments with more than one prototype per class.

Known classification results of the support vector machine vary between 15.5%

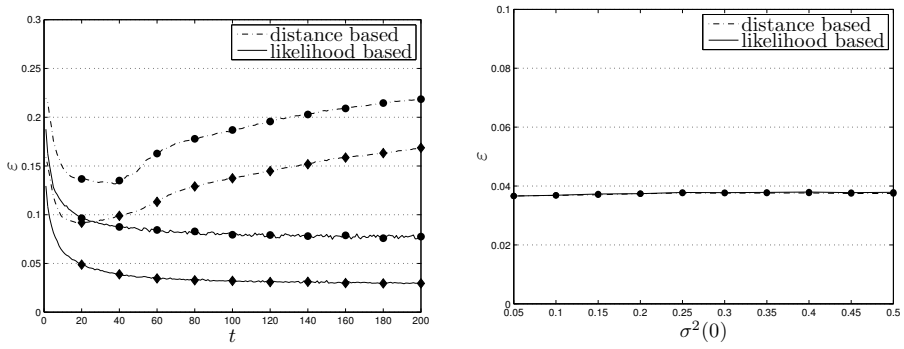


Figure 6.9: Pendigits *data set.* **Left:** *Mean error curves during training of local hyperparameters with one prototype per class and* $\sigma_j^2(0) = 0.3, \forall j$*; the curves are representative for all* $\sigma_j^2(0)$*. The symbols correspond to training error* (⧫)*, test error* (•)*.* **Right:** *Mean final test error after training of five prototypes per class and a global hyperparameter* σ^2 *as a function of the initial value* $\sigma^2(0)$*. Standard error bars would be smaller than the symbol size.*

and 8.5% mean error on the test sets depending on the kernel-function [1]. However, we would like to stress that our main interest in the experiments is related to the analysis of our methods in comparison to original RSLVQ. For this reason, further validation procedures to optimize the classifiers with respect to the number of prototypes or the incorporation of adaptive distance measures are not examined in this study.

Pendigits data set

This data set realizes a 10 class classification problem in a 16 dimensional feature space. The classification task consists in the recognition of the handwritten digits 0 to 9. The data was collected from 44 writers; each writer provided 250 samples. The samples of 33 writers form the training set, the data of the remaining 11 writers form the test set. As a preprocessing step, we normalize the data to zero mean and unit variance features. We perform all experiments with 10 different prototype initializations and present the averaged results in the following.

The first set of experiments deals with the adaptation of a global hyperparameter. We use one prototype per class and train the system with constant and adaptive σ^2 for 100 epochs. We apply the learning parameter settings $\alpha_1 = 0.001$, $\alpha_2 = 5 \cdot 10^{-4} \cdot \sigma^2(0)$ and $c = 0.01$. The initial values of the hyperparameter are selected from the interval $[0.05, 0.5]$.

[1] We thank U. Bodenhofer, Johannes Kepler University Linz, Austria, for providing the results.

The outcome confirms the observations made on the *Letter* data set: The initialization $\sigma^2(0)$ has no influence on the final value of the hyperparameter; it converges towards $\sigma^2_{final} \approx 0.3$ in all experiments (see Fig. 6.6, right). In consequence, the classification performance after training does not depend on $\sigma^2(0)$; the mean final test error is $\varepsilon_{test} = 7.7\%$ for all $\sigma^2(0)$. Learning with constant hyperparameter turns out to be very sensitive with respect to the value of the hyperparameter. However, training with constant $\sigma^2 = \sigma^2_{opt} = 0.15$ slightly outperforms training with adaptive σ^2; on average, the resulting classifiers achieve $\varepsilon_{test} = 7.3\%$ (see Fig. 6.5, right).

Training of local hyperparameters is continued for 200 epochs. As depicted in Fig. 6.9, left, the error curves for likelihood ratio based classification converge after ca. 100 epochs. However, the classification performance based on the Euclidean distance passes an optimum after ca. 40 epochs and degrades in the further course of training. Note that the curves do not reflect overfitting, since the mean error on the training sets increases simultaneously. After 40 sweeps through the training set, the likelihood ratio based classification performs $\approx 5\%$ better than the distance based approach. The final classification performance based on the likelihood ratio also slightly depends on the initialization, since the σ^2_j do not exactly approach the same value for different $\sigma^2_j(0)$ (see Fig. 6.8, right, for an example); in the examined interval of $\sigma^2_j(0)$, we observe final error rates between $\varepsilon_{test} = 7.6\%$ and $\varepsilon_{test} = 7.8\%$.

The decision rule does not influence the classification performance significantly, if we use a global hyperparameter and vary the number of prototypes. RSLVQ training with adaptive global σ^2 and five prototypes per class shows similar performance for both classification strategies (see Fig. 6.9, right). The hyperparameter converges towards $\sigma^2_{final} \approx 0.18$ for all $\sigma^2(0)$.

For comparison purposes, we refer to Tsang et al. (2006); Perfetti and Ricci (2006). Here, researches achieved between 1.8% and 2.2% mean test error using different variants of the SVM. Hence, our results are not yet competitive which was not the main objective of this study at the current state. In Sec. 6.4, we discuss possible approaches for future work to further improve the presented RSLVQ modifications.

6.4 Conclusion

We presented modifications of Robust Soft Learning Vector Quantization. We introduced a novel technique to treat the hyperparameter σ^2 and proposed an alternative decision rule different from the standard LVQ-approach of closest prototype classification. Furthermore, we extended the RSLVQ cost function with respect to vectorial input data.

As demonstrated in experiments, the classification accuracy of RSLVQ is highly

sensitive with respect to the correct choice of σ^2. However, an optimum choice of the hyperparameter is not possible based on statistical quantities directly from the data since its influence on the underlying discriminative objective function is not clear a priori. We proposed to adapt σ^2 according to the optimization of the likelihood ratio which takes the influence of the hyperparameter on the RSLVQ cost function into account. This approach made the algorithm very robust with respect to the hyperparameter and renders any trial and error search for an appropriate value unnecessary. Hence, the computational effort of a cross validation procedure can be avoided. In general, the model parameters do not exactly converge to the corresponding values of an underlying distribution given by mixtures of Gaussians. This fact can be explained by the influence of the parameters on the region of interest which influences the adaptation, on the one hand, and the discriminative power of the resulting model, on the other hand. As shown in the experiments, a simple automatic optimization provides a very robust scheme which converges to appropriate values almost independently on the initialization. Note that the parameter σ^2 is crucial for the success of the resulting classifier as demonstrated, e.g., in the mathematical investigation of the limit case for $\sigma^2 \to 0$ as provided in Biehl, Ghosh and Hammer (2007).

The proposed generalization of the RSLVQ cost function allows to account for insecure label information of training data in overlapping regions of different classes. Hence, uncertainty concerning class memberships can be integrated in the learning process of a nearest prototype classifier. The update scheme for the new algorithm is derived as a stochastic gradient of the generalized cost function and includes the RSLVQ updates as a special case. It was demonstrated based on artificial data that previous RSLVQ results arise out of Fuzzy RSLVQ under certain conditions.

Furthermore, we suggested the likelihood ratio of the RSLVQ cost function as a novel criterion for classification of the data. This concept follows naturally from the learning objective of the training procedure. In our experiments, the method turned out to be superior to distance based classification, in particular, if local hyperparameters are optimized for each prototype.

In future work, Fuzzy RSLVQ needs to be applied to more complex data sets. Real life applications will have to show to what extent the additional consideration of insecure label information influences the classification performance of a crisp classifier. Moreover, the question arises whether further generalization with respect to vectorial, adaptive prototype labels can be determined in a similar way. Vectorial system output allows to realize fuzzy class assignments. Thus, unsafe classification decisions can be realized, which is highly desirable, e.g. in medical applications.

A serious restriction of standard RSLVQ consists in the use of the Euclidean distance measure. In Sec. 3.3.3, the algorithm is extended with respect to adaptive

distance measures. We are currently combining metric adaptation in RSLVQ with the different approaches presented in this chapter, showing first promising results.

6.A Appendix: Derivatives of E_{FRSLVQ}

We compute the derivative of the cost function with respect to a general parameter $\Theta_i \neq \boldsymbol{\xi}$. We assume the densities $p(\boldsymbol{\xi}|j)$ to have the normalized exponential form $p(\boldsymbol{\xi}|j) = K(j) \cdot \exp f(\boldsymbol{\xi}, \boldsymbol{w}_j, \sigma_j^2, \boldsymbol{\lambda}_j)$, where $\boldsymbol{\lambda}_j$ is the vector of metric parameters.

$$
\begin{aligned}
&\frac{\partial \log \frac{p(\boldsymbol{\xi},\boldsymbol{y}|\mathbf{W})}{p(\boldsymbol{\xi}|\mathbf{W})}}{\partial \Theta_i} \\
&\quad = \frac{\partial \log p(\boldsymbol{\xi},\boldsymbol{y}|\mathbf{W})}{\partial \Theta_i} - \frac{\partial \log p(\boldsymbol{\xi}|\mathbf{W})}{\partial \Theta_i} \\
&\quad = \frac{1}{p(\boldsymbol{\xi},\boldsymbol{y}|\mathbf{W})} \underbrace{\frac{\partial p(\boldsymbol{\xi},\boldsymbol{y}|\mathbf{W})}{\partial \Theta_i}}_{(a)} - \frac{1}{p(\boldsymbol{\xi}|\mathbf{W})} \underbrace{\frac{\partial p(\boldsymbol{\xi}|\mathbf{W})}{\partial \Theta_i}}_{(b)} \\
&\quad = \frac{y^{c(\boldsymbol{w}_i)} P(i) \exp f(\boldsymbol{\xi}, \boldsymbol{w}_i, \sigma_i^2, \boldsymbol{\lambda}_i)}{p(\boldsymbol{\xi},\boldsymbol{y}|\mathbf{W})} \frac{\partial K(i)}{\partial \Theta_i} \\
&\qquad + \frac{y^{c(\boldsymbol{w}_i)} P(i) K(i) \exp f(\boldsymbol{\xi}, \boldsymbol{w}_i, \sigma_i^2, \boldsymbol{\lambda}_i)}{p(\boldsymbol{\xi},\boldsymbol{y}|\mathbf{W})} \frac{\partial f(\boldsymbol{\xi}, \boldsymbol{w}_i, \sigma_i^2, \boldsymbol{\lambda}_i)}{\partial \Theta_i} \\
&\qquad - \frac{y^{c(\boldsymbol{w}_i)} P(i) \exp f(\boldsymbol{\xi}, \boldsymbol{w}_i, \sigma_i^2, \boldsymbol{\lambda}_i)}{p(\boldsymbol{\xi}|\mathbf{W})} \frac{\partial K(i)}{\partial \Theta_i} \\
&\qquad - \frac{y^{c(\boldsymbol{w}_i)} P(i) K(i) \exp f(\boldsymbol{\xi}, \boldsymbol{w}_i, \sigma_i^2, \boldsymbol{\lambda}_i)}{p(\boldsymbol{\xi}|\mathbf{W})} \frac{\partial f(\boldsymbol{\xi}, \boldsymbol{w}_i, \sigma_i^2, \boldsymbol{\lambda}_i)}{\partial \Theta_i} \\
&\quad = \left(P_{\boldsymbol{y}}(i|\boldsymbol{\xi}) - P(i|\boldsymbol{\xi})\right) \left(\frac{1}{K(i)} \frac{\partial K(i)}{\partial \Theta_i} + \frac{\partial f(\boldsymbol{\xi}, \boldsymbol{w}_i, \sigma_i^2, \boldsymbol{\lambda}_i)}{\partial \Theta_i} \right)
\end{aligned}
\tag{6.11}
$$

with (a)

$$\begin{aligned}
&\frac{\partial p(\boldsymbol{\xi}, \boldsymbol{y}|\mathbf{W})}{\partial \Theta_i} \\
&\quad = \frac{\partial}{\partial \Theta_i} \left(\sum_{k=1}^{C} y^k \sum_{j=1}^{m} \delta_{k,c(\boldsymbol{w}_j)} P(j) p(\boldsymbol{\xi}|j) \right) \\
&\quad = \sum_{k=1}^{C} y^k \sum_{j=1}^{m} \delta_{k,c(\boldsymbol{w}_j)} P(j) \frac{\partial p(\boldsymbol{\xi}|j)}{\partial \Theta_i} \\
&\quad = \sum_{k=1}^{C} y^k \sum_{j=1}^{m} \delta_{k,c(\boldsymbol{w}_j)} P(j) \exp f(\boldsymbol{\xi}, \boldsymbol{w}_j, \sigma_j^2, \boldsymbol{\lambda}_j) \\
&\qquad\qquad \times \left(\frac{\partial K(j)}{\partial \Theta_i} + K(j) \frac{\partial f(\boldsymbol{\xi}, \boldsymbol{w}_j, \sigma_j^2, \boldsymbol{\lambda}_j)}{\partial \Theta_i} \right) \\
&\quad = y^{c(\boldsymbol{w}_i)} P(i) \exp f(\boldsymbol{\xi}, \boldsymbol{w}_i, \sigma_i^2, \boldsymbol{\lambda}_i) \left(\frac{\partial K(i)}{\partial \Theta_i} + K(i) \frac{\partial f(\boldsymbol{\xi}, \boldsymbol{w}_i, \sigma_i^2, \boldsymbol{\lambda}_i)}{\partial \Theta_i} \right)
\end{aligned}$$

and (b) given in Sec. 3.A.2.

$P_{\boldsymbol{y}}(i|\boldsymbol{\xi})$ is the assignment probability to component i within class $c(\boldsymbol{w}_i)$, taking the probability into account that $\boldsymbol{\xi}$ was generated by one of the components of class $c(\boldsymbol{w}_i)$

$$\begin{aligned}
P_{\boldsymbol{y}}(i|\boldsymbol{\xi}) &= \frac{\sum_k y^k \delta_{k,c(\boldsymbol{w}_i)} P(i) K(i) \exp f(\boldsymbol{\xi}, \boldsymbol{w}_i, \sigma_i^2, \boldsymbol{\lambda}_i)}{\sum_k y^k \sum_j \delta_{k,c(\boldsymbol{w}_j)} P(j) K(j) \exp f(\boldsymbol{\xi}, \boldsymbol{w}_j, \sigma_j^2, \boldsymbol{\lambda}_j)} \\
&= \frac{y^{c(\boldsymbol{w}_i)} P(i) K(i) \exp f(\boldsymbol{\xi}, \boldsymbol{w}_i, \sigma_i^2, \boldsymbol{\lambda}_i)}{\sum_k y^k \sum_j \delta_{k,c(\boldsymbol{w}_j)} P(j) K(j) \exp f(\boldsymbol{\xi}, \boldsymbol{w}_j, \sigma_j^2, \boldsymbol{\lambda}_j)}.
\end{aligned}$$

$P(i|\boldsymbol{\xi})$ depicts the probability the $\boldsymbol{\xi}$ is assigned to any component i of the mixture (see Sec. 3.A.2).

The derivative with respect to a global parameter, e.g. a global hyperparameter $\sigma^2 = \sigma_j^2$ for all j can be derived thereof by summation.

Chapter 7

Conclusion

7.1 Summary

The thesis presents different concepts to improve the performance of LVQ algorithms. One issue refers to metric learning in LVQ, i.e., the optimization of the employed distance measure for a specific application. A novel parameterized distance measure is proposed which can be adapted to the training data by means of any LVQ algorithm. Furthermore, three approaches to modify Robust Soft LVQ are introduced.

After providing the required background to Learning Vector Quantization in chapter 2, the novel distance measure is introduced in chapter 3. The Euclidean distance is extended by a full matrix of adaptive weight values. This approach generalizes the popular concept of relevance learning. The diagonal elements of the adaptive matrix correspond to an explicit feature weighting. The off-diagonal elements additionally weight combinations of features. Hence, correlations can be taken into account to evaluate the similarity between prototypes and feature vectors. Equivalently, this metric corresponds to the squared Euclidean distance after performing a linear transformation of data and prototypes to a new feature space. Full rank matrices can be learned as well as distance measures which are restricted to a smaller number of features. Moreover, global, as well as class-wise or prototype-specific metrics can be implemented. We derive the learning rules for matrix learning in Generalized LVQ and Robust Soft LVQ. Experiments on artificial and benchmark real life data sets illustrated the new technique in practice. The applications demonstrate that matrix learning is advantageous in two respects: The approach is clearly beneficial with respect to classification accuracy compared to the use of the Euclidean distance or the weighted Euclidean distance. This holds especially for local matrix adaptation in case of multi-class problems. Furthermore, the additional parameters improve the interpretability of the final model. Matrix parameters reveal the importance of the input dimensions for the diagonal elements and the importance of correlations for the off-diagonal elements. Additionally, the experiments

show that the two training algorithms display different learning dynamics and different characteristics of the resulting model.

In chapter 4, we present a regularization technique to influence the convergence behavior of matrix learning. The proposed method prevents the training algorithm from yielding low rank distance measures. As it is frequently observed, the update rules tend to select a single or very few directions in feature space. We propose a regularization term which favors balanced eigenvalue profiles of the relevance matrix. The technique can be applied in combination with any matrix learning scheme. We demonstrate the usefulness by means of matrix learning in GLVQ. Beside the desired effect on the convergence behavior, the technique turns out to be beneficial to prevent over-fitting effects and numerical instabilities during training.

Theoretical aspects of matrix learning in LVQ are investigated in chapter 5. In particular, we research the convergence behaviour in terms local matrix adaptation in LVQ1. Under simplifying model assumptions, it is shown that the updates lead to the selection of single directions in feature space. This direction is determined by the statistical properties of the data assigned to the respective prototype. Furthermore, we investigate the effect of the proposed regularization approach in this framework. We show that the method eases the tendency of the learning algorithm to strongly reduce the number of dimensions. The derivations are verified by practical experiments.

Chapter 6 presents three approaches to extend Robust Soft LVQ. The study refers to the original version of the algorithm based on the squared Euclidean distance. First, we propose to adapt the algorithm's hyperparameter in the course of training based on the gradient information of the RSLVQ cost function. Our experiments depict that hyperparameter learning makes the algorithm very robust with respect to the initial choice of the hyperparameter. Moreover, local hyperparameters can be learned for every prototype individually. Furthermore, we present an alternative decision rule for RSLVQ classifiers: Training the model parameters maximizes the underlying cost function which is defined in terms of a likelihood ratio. Against this background, we propose to assign a sample in the working phase to the class of highest likelihood ratio. Contrary to the distance based classification rule, this approach naturally follows the objective of RSLVQ training. The following practical examples show that the novel decision rule outperforms the distance based approach, if the method is combined with local, adaptive hyperparameters. Finally, we derive the generalization of the RSLVQ cost function with respect to vectorial class labels of the input data. We demonstrate on artificial data set that RSLVQ is a special case of the novel fuzzy algorithm.

7.2 Future work

This work could be extended in several directions. In particular, we consider the following topics for future research:

- Throughout the thesis, we apply the novel algorithms to benchmark data sets. Future work will address the application to more sophisticated data sets. Due to the interpretability and the easy handling of missing values, LVQ is especially attractive for interdisciplinary applications in the medical or biological domain. Relevance learning has already proven to be valuable for the analysis of such data, e.g. for biomarker detection. Matrix learning has the potential to further advance this work. Currently, we are working on a diagnosis system for adrenal cancer using the algorithms presented in chapter 3. The experiments show promising results which will be presented in a forthcoming publication.

- Future work will also combine the proposed extensions of Robust Soft LVQ (Sec. 6.2) with matrix learning in RSLVQ (Sec. 3.3.3)

- Several learning problems introduced in this thesis are formulated in terms of the optimization of a cost function, and the optimization is implemented in terms a stochastic gradient descent or ascent. Alternatively, advanced optimization algorithms could be used to train the model parameters, e.g. line search methods or the conjugate gradient algorithm (Bishop, 1995). Although these methods are computationally more expensive, the approach could eliminate the sensitivity of the learning process with respect to initialization and learning parameter. Advanced optimization techniques could make to learning dynamics more robust and reduce the required number of learning steps.

- Finally, continuing the work presented in Sec. 6.2.3, we plan to further generalize the RSLVQ cost function with respect to vectorial, adaptive class labels for the prototypes. This natural extension of Fuzzy RSLVQ allows to realize insecure classification results, which is highly desirable, e.g. in medical applications.

Samenvatting

Dit proefschrift beschrijft verschillende concepten voor het verbeteren van LVQ algoritmen. Een van de behandelde punten is het aanleren van een metriek binnen LVQ, dat wil zeggen, het optimaliseren van de gebruikte afstandsmaat voor een specifieke toepassing. Een nieuwe geparametriseerde afstandsmaat wordt voorgesteld, welke door ieder LVQ-algoritme aangepast kan worden aan de traindata. Ook zullen drie verschillende benaderingen om Robust Soft LVQ aan te passen geïntroduceerd worden.

Na het geven van de benodigde achtergrond van LVQ in hoofdstuk 2 zal een nieuwe afstandsmaat geïntroduceerd worden in hoofdstuk 3. De Euclidische afstand wordt uitgebreid met een volledige matrix van adaptieve gewichten. Deze aanpak generaliseert het populaire concept van het leren van relevantie. De diagonale elementen van de adaptieve matrix corresponderen met het expliciet wegen van features, terwijl de buitendiagonale elementen gewicht toevoegen van combinaties van features. Zodoende kunnen correlaties in acht worden genomen bij het evalueren van de gelijkheidswaarde tussen prototypen en featurevectoren. Deze metriek is equivalent aan de gekwadrateerde Euclidische afstand na een lineaire transformatie van data en prototypen naar een nieuwe featureruimte. Zowel matrices met een volledige rang als afstandsmaten welke gelimiteerd zijn tot een klein aantal features kunnen aangeleerd worden. Daar bovenop kan zowel een globaal als een klassegewijs of prototype-specifiek metriek geïmplementeerd worden. We leiden de regels af voor het aanleren van matrices in Generalized LVQ en Robust Soft LVQ. Experimenten met kunstmatige datasets en reële referentiedatasets illustreren de nieuwe techniek in de praktijk. De toepassingen demonstreren dat het aanleren van de matrix is op twee punten profijtelijk: de aanpak heeft een duidelijke meerwaarde wanneer de accuraatheid van de classificatie in beschouwing wordt

genomen en wordt vergeleken met het gebruik van de Euclidische afstand of de gewogen Euclidische afstand. Dit geldt in het speciaal voor lokale-matrixadaptatie bij multi-klasseproblemen. Daarbovenop verbeteren de toegevoegde parameters de interpreteerbaarheid van het uiteindelijke model. Matrixparameters onthullen de belangrijke rol van de invoerdimensies op de diagonaalelementen en de belangrijke rol van correlaties op de buitendiagonale elementen. De experimenten leren ons alsmede dat de twee leeralgoritmen verschillen in leerdynamica en de karakteristieken van het resulterende model.

In hoofdstuk 4 presenteren we een regularisatietechniek om het convergentiegedrag van matrixleren te beïnvloeden. De voorgestelde methode voorkomt dat het leeralgoritme laagrangse afstandsmaten produceert. Zoals vaak wordt geobserveerd, hebben de aanpassingsregels de neiging om een enkele of een erg klein aantal richtingen in featureruimte te kiezen. We introduceren een regularisatieterm welke gebalanceerde eigenwaardeprofielen verkiest van de relevantiematrix. De techniek kan toegepast worden in combinatie met ieder matrixleerschema. We demonstreren haar bruikbaarheid door matrixleren met GLVQ. Naast het gewenste effect op het convergentiegedrag, blijkt de techniek profitabel om overspecialisatie-effecten en numerieke instabiliteiten te voorkomen tijdens het aanleren.

Theoretische aspecten van het matrixleren in LVQ worden onderzocht in hoofdstuk 5. In het bijzonder onderzoeken we het convergentiegedrag in termen van lokale-matrixadaptatie in LVQ1. Bij gesimplificeerde modelaannames wordt aangetoond dat de aanpassingen er toe leiden dat een enkele richting in featureruimte gekozen wordt. Deze richting wordt bepaald door de statistische eigenschappen van de data welke toegekend is aan het respectievelijke prototype.

Hoofdstuk 6 presenteert drie verschillende werkwijzes om Robust Soft LVQ uit te breiden. De studie refereert naar de oorspronkelijke versie van het algoritme dat is gebaseerd op de gekwadrateerde Euclidische afstand. Allereerst stellen we voor om tijdens het trainen van het algoritme, gebaseerd op de gradiëntinformatie van de RSLVQ-kostenfunctie, zijn hyperparameter te wijzigen. Onze experimenten bevestigen dat het leren van de hyperparameter zorgt voor een zeer robuust algoritme ten aanzien van de initiële keus van de hyperparameter. Ook kan voor ieder prototype individueel een lokale hyperparameter aangeleerd worden. Aansluitend presenteren we een alternatieve besluitregel voor RSLVQ-klassificatie: het trainen van de modelparameters maximaliseert de onderliggen kostenfunctie, welke gedefinieerd is door een waarschijnlijkheidsratio. Met deze achtergrond stellen we voor om een nieuw monster tijdens de werkfase toe te kennen aan de klasse met de hoogste waarschijnlijkheidsratio. In tegenstelling tot de afstandgebaseerde klassificatieregel volgt deze aanpak van nature het doel van RSLVQ-training. De hieropvolgende praktische voorbeelden laten zien dat de nieuwe beslissingsregel beter presteert dan

de afstandgebaseerde aanpak, wanneer de methode gecombineerd wordt met lokale adaptieve hyperparameters. Tot slot leiden we de generalisatie van de RSLVQ-kostenfunctie af ten aanzien van de vectoriale klassebenoeming van de invoerdata. Op kunstmatige data demonstreren we dat RSLVQ een special situatie is van het nieuwe fuzzy algoritme.

Bibliography

Arnonkijpanich, B., Hammer, B., Hasenfuss, A. and Lursinap, A.: 2008, Matrix learning for topographic neural maps, *International Conference on Artificial Neural Networks*, Prague, Czech Republic, pp. 572–582.

Bibliography on the Self-Organizing Map (SOM) and Learning Vector Quantization (LVQ): 2002, Neural Networks Research Centre, Helskinki University of Technology.

Biehl, M., Breitling, R. and Li, Y.: 2007, Analysis of tiling microarray data by learning vector quantization and relevance learning, *International Conference on Intelligent Data Engineering and Automated Learning*, Springer LNCS, Birmingham, UK, pp. 880–889.

Biehl, M., Ghosh, A. and Hammer, B.: 2007, Dynamics and generalization ability of LVQ algorithms, *Journal of Machine Learning Research* **8**, 323–360.

Biehl, M., Hammer, B., Schleif, F.-M., Schneider, P. and Villmann, T.: 2009, Stationarity of relevance matrix learning vector quantization, *Technical Report MLR-01-2009*, University of Leipzig.

Biehl, M., Hammer, B., Verleysen, M. and Villmann, T. (eds): 2009, *Similarity based clustering - recent developments and biomedical applications*, Vol. 5400 of *Lecture Notes in Artificial Intelligence*, Springer.

Biehl, M., Pasma, P., Pijl, M. and Petkov, N.: 2006, Classification of boar sperm head images using learning vector quantization, *in* M. Verleysen (ed.), *European Symposium on Artificial Neural Networks*, Bruges, Belgium, pp. 545–550.

Biehl, M., Schneider, P., Hammer, B., Schleif, F.-M. and Villmann, T.: submitted, 2010, Stationarity of matrix updates in relevance learning vector quantization.

Bishop, C. M.: 1995, *Neural Networks for Pattern Recognition*, 1 edn, Oxford University Press.

Bishop, C. M.: 2007, *Pattern Recognition and Machine Learning*, 1 edn, Springer.

Boehm, W. and Prautzsch, H.: 1993, *Numerical Methods*, Vieweg.

Bojer, T., Hammer, B., Schunk, D. and von Toschanowitz, K. T.: 2001, Relevance determination in learning vector quantization, *in* M. Verleysen (ed.), *European Symposium on Artificial Neural Networks*, Bruges, Belgium, pp. 271–276.

Cover, T. and Hart, P.: 1967, Nearest neighbor pattern classification, *IEEE Transactions on Information Theory* **13**(1), 21–27.

Crammer, K., Gilad-Bachrach, R., Navot, A. and Tishby, A.: 2003, Margin analysis of the lvq algorithm, *Advances in Neural Information Processing Systems*, Vol. 15, MIT Press, Cambridge, MA, USA, pp. 462–469.

Darken, C., Chang, J., Z, J. C. and Moody, J.: 1992, Learning rate schedules for faster stochastic gradient search, *Neural Networks for Signal Processing 2 - Proceedings of the 1992 IEEE Workshop*, IEEE Press.

Duda, R., Hart, P. and Stork, D.: 2000, *Pattern Classification*, second edn, Wiley-Interscience.

Gath, I. and Geva, A. B.: 1989, Unsupervised optimal fuzzy clustering, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **11**(7), 773–780.

Ghosh, A., Biehl, M. and Hammer, B.: 2006, Performance analysis of lvq algorithms: a statistical physics approach, *Neural Networks* **19**(6), 817–829.

Gustafson, E. and Kessel, W.: 1979, Fuzzy clustering with a fuzzy covariance matrix, *IEEE Conference on Decision and Control*, San Diego, CA, USA, pp. 761–766.

Hammer, B., Schleif, F.-M. and Villmann, T.: 2005, On the generalization ability of prototype-based classifiers with local relevance determination, *Technical Report Ifi-05-14*, Clausthal University of Technology.

Hammer, B., Strickert, M. and Villmann, T.: 2004, Prototype based recognition of splice sites, *Bioinformatic using Computational Intelligence Paradigms*, Springer-Verlag, pp. 25–56.

Hammer, B., Strickert, M. and Villmann, T.: 2005a, On the generalization ability of GRLVQ networks, *Neural Processing Letters* **21**(2), 109–120.

Hammer, B., Strickert, M. and Villmann, T.: 2005b, Supervised neural gas with general similarity measure, *Neural Processing Letters* **21**(1), 21–44.

Hammer, B. and Villmann, T.: 2002, Generalized relevance learning vector quantization, *Neural Networks* **15**(8-9), 1059–1068.

Kaski, S.: 2001, Principle of learning metrics for exploratory data analysis, *Neural Networks for Signal Processing XI, Proceedings of the 2001 IEEE Signal Processing Society Workshop*, IEEE, pp. 53–62.

Kietzmann, T. C., Lange, S. and Riedmiller, M.: 2008, Incremental grlvq: Learning relevant features for 3d object recognition, *Neurocomputing* **71**(13-15), 2868–2879.

Kohonen, T.: 1986, Learning vector quantization for pattern recognition, *Technical Report TKK-F-A601*, Helsinki Univeristy of Technology, Espoo, Finland.

Kohonen, T.: 1990, Improved versions of learning vector quantization, *International Joint Conference on Neural Networks*, Vol. 1, pp. 545–550.

Kohonen, T.: 1997, *Self-Organizing Maps*, second edn, Springer, Berlin, Heidelberg.

Kohonen, T.: 1998, Learning vector quantization, *The handbook of brain theory and neural networks*, MIT Press, Cambridge, MA, USA, pp. 537–540.

Kusumoputro, B. and Budiarto, H.: 1999, Improvement of artificial odor discrimination system using fuzzy-lvq neural network, *International Conference on Computational Intelligence and Multimedia Applications*, IEEE Computer Society, Los Alamitos, CA, USA, pp. 474–478.

Martinetz, T. and Schulten, K.: 1991, A "neural-gas" network learns topologies, *Artificial Neural Networks* **I**, 397–402.

Mendenhall, M. and Merényi, E.: 2006, Generalized relevance learning vector quantization for classification driven feature extraction from hyperspectral data, *Proceedings of ASPRS 2006 Annual Conference and Technology Exhibition*, p. 8.

Mwebaze, E., Schneider, P., Schleif, F.-M., Haase, S., Villmann, T. and Biehl, M.: 2010, Divergence based learning vector quantization, *in* M. Verleysen (ed.), *European Symposium on Artificial Neural Networks*, Bruges, Belgium, pp. 247–252.

Newman, D. J., Hettich, S., Blake, C. L. and Merz, C. J.: 1998, Uci repository of machine learning databases, `http://archive.ics.uci.edu/ml/`.

Ong, C., A. Smola, A. and Williamson, R.: 2005, Learning the kernel with hyperkernels, *Journal of Machine Learning Research* **6**, 1043–1071.

Perfetti, R. and Ricci, E.: 2006, Reduced complexity rbf classifiers with support vector centres and dynamic decay adjustment, *Neurocomputing* **69**(16-18), 2446–2450.

Petersen, K. B. and Pedersen, M. S.: 2008, The matrix cookbook, `http://matrixcookbook.com`.

Prudent, Y. and Ennaji, A.: 2005, A k nearest classifier design, *Electronic Letters on Computer Vision and Image Analysis* **5**(2), 58–71.

Sato, A. and Yamada, K.: 1996, Generalized learning vector quantization, *in* M. C. M. D. S. Touretzky and M. E. Hasselmo (eds), *Advances in Neural Information Processing Systems*, Vol. 8, MIT Press, Cambridge, MA, USA, pp. 423–429.

Sato, A. and Yamada, K.: 1998, An analysis of convergence in generalized lvq, *in* L. Niklasson, M. Bodén and T. Ziemke (eds), *Proceedings of the International Conference on Artificial Neural Networks*, Springer, pp. 170–176.

Schneider, P., Biehl, M. and Hammer, B.: 2007, Relevance matrices in lvq, *in* M. Verleysen (ed.), *European Symposium on Artificial Neural Networks*, Bruges, Belgium, pp. 37–42.

Schneider, P., Biehl, M. and Hammer, B.: 2009a, Adaptive relevance matrices in learning vector quantization, *Neural Computation* **21**(12), 3532–3561.

Schneider, P., Biehl, M. and Hammer, B.: 2009b, Distance learning in discriminative vector quantization, *Neural Computation* **21**(10), 2942–2969.

Schneider, P., Bunte, K., Stiekema, H., Hammer, B., Villmann, T. and Biehl, M.: 2010, Regularization in matrix relevance learning, *IEEE Transactions on Neural Networks* **21**(5), 831–840.

Schneider, P., Schleif, F.-M., Villmann, T. and Biehl, M.: 2008, Generalized matrix learning vector quantizer for the analysis of spectral data, *in* M. Verleysen (ed.), *European Symposium on Artificial Neural Networks*, Bruges, Belgium, pp. 451–456.

Seo, S., Bode, M. and Obermayer, K.: 2003, Soft nearest prototype classification, *IEEE Transactions on Neural Networks* **14**(2), 390–398.

Seo, S. and Obermayer, K.: 2003, Soft learning vector quantization, *Neural Computation* **15**(7), 1589–1604.

Seo, S. and Obermayer, K.: 2006, Dynamic hyper parameter scaling method for lvq algorithms, *International Joint Conference on Neural Networks*, Vancouver, CA.

Shalev-Schwartz, S., Singer, Y. and Ng, A.: 2004, Online and batch learning of pseudo-metrics, *Proceedings of the 21st International Conference on Machine Learning*, ACM, New York, USA, p. 94.

Strickert, M., Witzel, K., Mock, H.-P., Schleif, F.-M. and Villmann, T.: 2007, Supervised attribute relevance determination for protein identification in stress experiments, *Machine Learning in Systems Biology*, pp. 81–86.

Tamura, H. and Tanno, K.: 2008, Midpoint-validation method for support vector machine classification, *IEICE - Transactions on Information Systems* **E91-D**(7), 2095–2098.

Thiel, C., Sonntag, B. and Schwenker, F.: 2008, Experiments with supervised fuzzy lvq, *in* L. Prevost, S. Marinai and F. Schwenker (eds), *Artificial Neural Networks in Pattern Recognition*, Vol. 5064 of *Lecture Notes in Computer Science*, Springer, pp. 125–132.

Tsang, I. W., Kocsor, A. and Kwok, J. T.: 2006, Diversified svm ensembles for large data sets, *Machine Learning: ECML 2006*, Vol. 4212 of *Lecture Notes in Computer Science*, Springer, pp. 792–800.

Weinberger, K., Blitzer, J. and Saul, L.: 2006, Distance metric learning for large margin nearest neighbor classification, *in* Y. Weiss, B. Schölkopf and J. Platt (eds), *Advances in Neural Information Processing Systems*, MIT Press, Cambridge, MA, USA, pp. 1473–1480.

Wu, K.-L. and Yang, M.-S.: 2003, A fuzzy-soft learning vector quantization, *Neurocomputing* **55**(3-4), 681–697.